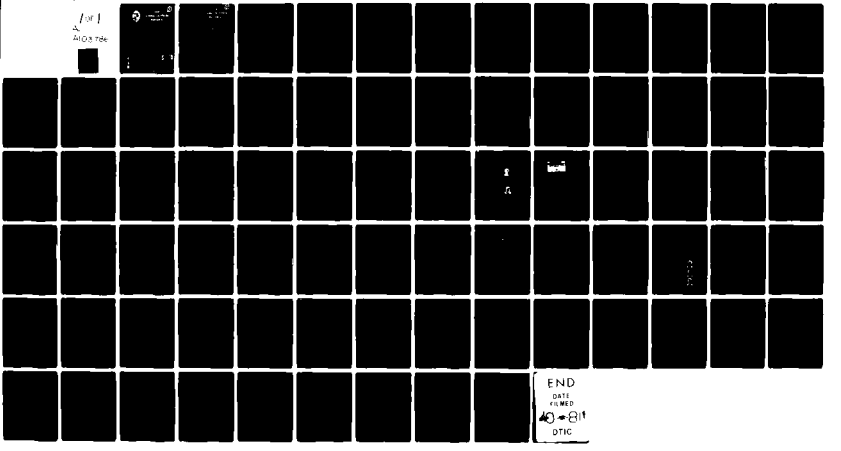
AD-A103 786 JOHNS HOPKINS UNIV BALTIMORE MD DEPT OF ELECTRICAL --ETC F/G 9/5
RELIABLE SWITCHING CIRCUITS FOR NAVAL COMMUNICATIONS,(U)
1981 G M MASSON N00014-75-C-1196
UNCLASSIFIED NL



LEVEL II

1

THE JOHNS HOPKINS UNIVERSITY

AD A103786

ELECTRICAL ENGINEERING DEPARTMENT

REPORT

(6) RELIABLE SWITCHING CIRCUITS FOR NAVAL COMMUNICATIONS,

BY

(10) GERALD M. MASSON
DEPARTMENT OF ELECTRICAL ENGINEERING
G.W.C. WHITING SCHOOL OF ENGINEERING
THE JOHNS HOPKINS UNIVERSITY
BALTIMORE, MARYLAND 21218

(11) 1981

(12) 77

DTIC FILE COPY

DTIC ELECTE SEP 4 1981 S D

SUPPORTED BY (15)
OFFICE OF NAVAL RESEARCH CONTRACT N00014-75-C1196
CODE 430 NR 048-630

400464

81 8 07 005

(1)

THE JOHNS HOPKINS UNIVERSITY

ELECTRICAL ENGINEERING DEPARTMENT

REPORT

RELIABLE SWITCHING CIRCUITS FOR NAVAL COMMUNICATIONS

BY

GERALD M. MASSON
DEPARTMENT OF ELECTRICAL ENGINEERING
G.W.C. WHITING SCHOOL OF ENGINEERING
THE JOHNS HOPKINS UNIVERSITY
BALTIMORE, MARYLAND 21218

SUPPORTED BY

OFFICE OF NAVAL RESEARCH CONTRACT N00014-75-C1196
CODE 430 NR 048-630

new

Resolution-Oriented Fault Interrelationships in Combinational Logic Networks

VINOD K. AGARWAL AND GERALD M. MASSON

Abstract—This correspondence considers fault resolution as a process of applying a sequence of input vectors, called tests, to a combinational logic network in order to resolve an existing fault situation *from* within a given *master* set of faults. A functional approach based upon an extension of the well-known Boolean difference concept to fault dependent situations is described. The

Manuscript received February 23, 1976; revised June 1, 1976. This work was supported in part by Office of Naval Research Contract N00014-75-C-1196.

The authors are with the Department of Electrical Engineering, The Johns Hopkins University, Baltimore, MD 21218.

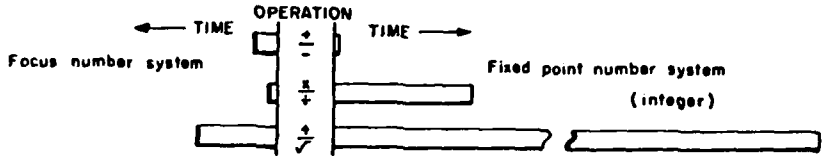


Fig. 2. Comparison of execution times between the focus number system and an integer system.

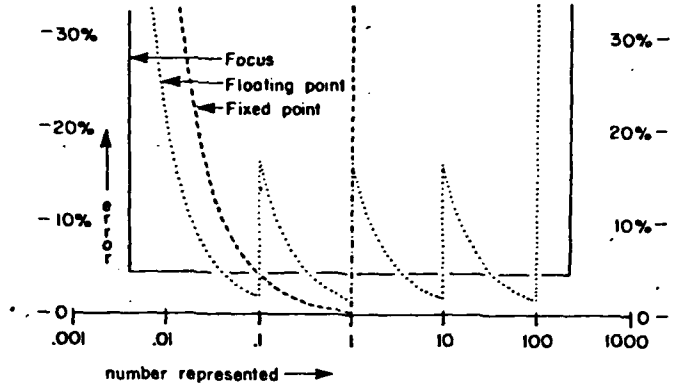


Fig. 3. Comparison of accuracy among the focus, fixed-point, and floating-point number systems.

raising a number to a noninteger power is performed essentially like a single multiplication in the prior art, and the wide range of possible states can represent very large or very small answers. The difficult but useful square-root function is extracted in focus simply by shifting right 1 bit, then adding 0 1000.000.

V. Comparison of Focus to the Prior Art

Fig. 2 is a qualitative comparison of execution times between the focus-number system and an integer system requiring software multiplication. Exact execution times are a function of many things (for example, inclusion or exclusion of optional steps and facileness of the machine language); however, overall it is expected that the focus-number-system algorithms will perform a mixture of calculations about five times faster in a microcomputer than a prior-art fixed-point (integer) system using software multiplication. This estimated number does not include the time savings expected by using single instead of double precision, as is allowed by focus in certain applications. Software floating-point times are not shown, but this would be slower than fixed-point arithmetic.

Fig. 3 is a quantitative comparison of accuracy among the focus, fixed-point (integer), and floating-point number systems, in which all are normalized to 8-bit bipolar systems. The error figure given is the maximum relative error in the region around any given number represented. This is half the step size, and is $\sqrt{3}$ times the root-mean square error in that region. In this figure the fixed-point system is seen to be the most accurate system in a narrow range between very large and very small numbers. It is thus useful in processing a well-behaved signal for which the virtual ground concept of focus control or the wide range of the focus system are not required, and where speed is not important. Integer arithmetic is also superior in certain operations involving chain addition, such as a program counter. The floating-point system inefficiently crowds states when the mantissa is large, leaving sparsely covered regions of high error scattered across the number line. Base ten was chosen for comparison. A larger base would result in wider oscillations and a smaller base would result in a higher average error due to redundant states; for example, base two would have a 50-percent state redundancy resulting in an average near the 10-percent error line. Floating point, although very slow without a costly arithmetic unit, is attractive in large computers of over 16 bits for which the lookup table of focus becomes impractically long. If a direct hardware implementation of the focus table is found, floating-point arithmetic will be seriously challenged in all applications.

VI. Conclusion

It is concluded that, for a fixed word length, Focus is superior in accuracy to floating-point arithmetic in all cases and to fixed point (integer) in many cases, especially including control systems for which the sensing control signals can be reduced by differential inputs or other analog means into forms that approach zero as the system approaches the desired controlled state.

References

[1] T. C. Bartee, *Digital Computer Fundamentals*. New York: McGraw-Hill, 1972, ch. 3.
[2] G. A. Maley and M. F. Heilweil, *Introduction to Digital Computers*. Englewood Cliffs, NJ: Prentice-Hall, 1968, ch. 7.
[3] S. C. Lee, *Digital Circuits and Logic Design*. Englewood Cliffs, NJ: Prentice-Hall, 1976, ch. 2.

Resolution-Oriented Fault Interrelationships in Combinational Logic Networks

VINOD K. AGARWAL AND GERALD M. MASSON

***Abstract*—This correspondence considers fault resolution as a process of applying a sequence of input vectors, called tests, to a combinational logic network in order to resolve an existing fault situation from within a given master set of faults. A functional approach based upon an extension of the well-known Boolean difference concept to fault dependent situations is described. The**

Manuscript received February 23, 1976; revised June 1, 1976. This work was supported in part by Office of Naval Research Contract N00014-75-C-1196.

The authors are with the Department of Electrical Engineering, The Johns Hopkins University, Baltimore, MD 21218.

$$T(F_M) = \{X \mid Z[X;Y_M] \oplus Z[X;\alpha_M] = 1\}. \quad (1)$$

In words, for every $X \in T(F_M)$, $Z[X;Y_M]$ is different than the output of the faulty network, $Z[X;\alpha_M]$. Thus, when any test $X \in T(F_M)$ is applied to a network, if the Boolean output of the network is different than the Boolean value of the output function describing the fault-free network, we say the network has *failed* the test; otherwise we say the network has *passed* the test.

Clearly, the test set $T(F_M)$ is based on comparing the operation of a faulty network with that of a fault-free network. However, in order to characterize the fault interrelationships germane to the fault resolution process, it will be necessary to compare the operations of the network under two fault conditions. Thus, assume that fault condition F_J is known to exist in N, and we are to determine a complete test set for detecting F_I in network N with F_J present. We will refer to such a test set as a fault dependent test set, and denote it as $T(F_I|F_J)$. It should be clear that

$$T(F_I|F_J) = \{X \mid Z[X;Y_I,\alpha_J] \oplus Z[X;\alpha_I,\alpha_J] = 1\}. \quad (2)$$

Note that such a test is based on a fault dependent Boolean difference which only gives valid results for the network if, indeed, this assumed fault F_J exists in the network. Also note that in applying a fault dependent test, the operation of fault-free network is not of concern, and the conclusion as to whether or not the network has failed or passed the test must take this into account. Our convention for this will be the following. For a test $X_1 \in T(F_I|F_J)$, let the logical value of the Boolean expression $Z[X_1;Y_I,\alpha_J]$ be a, $a \in \{0,1\}$. Now, if the output of the network when this test is applied is $\bar{a}$, we say the network has *failed* the test; otherwise we say the network has *passed* the test.

A fundamental interrelationship which exists between fault dependent test sets and standard test sets is described in the following theorem.

Theorem 1:

$$T(F_I|F_J) = T(F_J) \oplus T(F_I \cup F_J).$$

Proof: Since $Z[X;Y_I,Y_J] \oplus Z[X;Y_I,Y_J] = 0$, it is clear from (2) that $T(F_I|F_J) = \{X \mid (Z[X;Y_I,\alpha_J] \oplus Z[X;\alpha_I,\alpha_J]) \oplus (Z[X;Y_I,Y_J] \oplus Z[X;Y_I,Y_J]) = 1\}$.

However, by simply rearranging the above equation, we obtain

$$T(F_I|F_J) = \{X \mid (Z[X;Y_I,\alpha_J] \oplus Z[X;Y_I,Y_J]) \oplus (Z[X;\alpha_I,\alpha_J] \oplus Z[X;Y_I,Y_J]) = 1\}$$

or

$$T(F_I|F_J) = T(F_J) \oplus T(F_I \cup F_J). \quad \text{Q.E.D.}$$

Example 1[1]*:* To illustrate the above, consider the network of Fig. 1. Suppose $M = \{1,2,3,5\}$. Let $I = \{1,2\}$ and $\alpha_I = \{\alpha_1,\alpha_2\} = \{1,1\}$, and let $J = \{3,5\}$ and $\alpha_J = \{\alpha_3,\alpha_5\} = \{0,1\}$. It can be seen that $Z[X;Y_M] = (Y_3x_4 + Y_2)Y_1 + \overline{(Y_3x_4 + Y_2)}(Y_5 + \overline{x_4})$. Suppose now that F_J is known to exist in the network, and we want to determine whether F_I also exists. By using (2), we get

$$T(F_I|F_J) = \{X \mid Z[X;Y_I,\alpha_J] \oplus Z[X;\alpha_I,\alpha_J] = 1\} = \{X \mid \bar{x}_1\bar{x}_2 = 1\}.$$

We now choose, say, $X_1 = (0000) \in T(F_I|F_J)$ and then determine that $Z[X_1;Y_I,\alpha_J] = 0$. Now if we apply X_1 to the circuit containing F_J, the output would be a 1 if F_I were present and a 0 otherwise. Also note that since $T(F_I \cup F_J) = \{X \mid \bar{x}_1\bar{x}_2 + x_1x_2x_3x_4 + \bar{x}_2\bar{x}_3x_4 = 1\}$ and $T(F_J) = \{X \mid x_1\bar{x}_2\bar{x}_3x_4 + x_1x_2x_3x_4 = 1\}$, by Theorem 1, we have $T(F_I|F_J) = T(F_J) \oplus T(F_I \cup F_J) = \{X \mid \bar{x}_1\bar{x}_2 = 1\}$.

[1] Fig. 1 is taken from Ku and Masson [1].

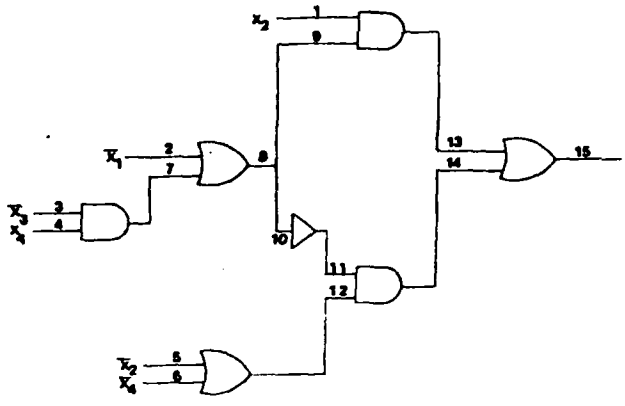


Fig. 1. $Z[X;\emptyset] = \bar{x}_1x_2 + x_2\bar{x}_3x_4 + x_1\bar{x}_2x_3 + x_1\bar{x}_4$.

The fault dependent test set will be seen in the following section to be fundamental to the characterization of the resolution-oriented fault interrelationships. Theorem 1 then indicates a straightforward means by which standard testing information can be used to consider the existence of such fault interrelationships in actual situations. However, Theorem 1 also gives significant insight into the concept of a fault dependent test, as it indicates that there are really two types of fault dependent tests: one type of which seems reasonably satisfying intuitively, the other of which seems significantly less so. To illustrate this, let $X_1 \in T(F_I|F_J)$ and let the output of the fault-free network with X_1 applied to it be $Z[X_1;Y_I,Y_J] = 1$. Now, we will consider the following two cases.

Case 1: Let X_1 be such that $Z[X_1;Y_I,\alpha_J] = 1$ and

$$Z[X_1;\alpha_I,\alpha_J] = 0.$$

Since X_1 truly tests [in the sense of (1)] for $F_I \cup F_J$ while not testing for F_J, this test seems to agree with what could be naturally accepted as a fault dependent test for F_I in the presence of F_J.

Case 2: Let X_1 be such that $Z[X_1;Y_I,\alpha_J] = 0$ and

$$Z[X_1;\alpha_I,\alpha_J] = 1.$$

However, in this case, since X_1 is a test which does not take into account the effect of the fault F_I relative to the fault-free network, such a test is intuitively less satisfying. It is this interesting aspect of a fault dependent test, however, which makes it a powerful tool in fault resolution considerations. Note that by Theorem 1, any fault dependent test is always one of the above two types.

IV. Resolution-Oriented Fault Interrelationships

There exists an inherent limit to the degree to which a fault situation can be resolved within a master set, and this limit depends upon the fault interrelationships which exist among the faults within this set and the knowledge regarding the existing fault situation which has thus far been attained. By means of the results of the previous section, we are now able to characterize pertinent fault interrelationships which describe these limitations. To do so, we will consider the master set of faults F_M, and then describe the fault interrelationships between subsets of faults within this master set, say, F_I and F_J. Note that it is not necessary that $F_I \cup F_J = F_M$.

Our characterizations of these resolution-oriented fault interrelationships will be given by means of the fault dependent Boolean difference. Hence, we will immediately have algebraic formulas with which to verify whether these conditions actually exist within a network for some subsets of a given master set of faults.

Definition 1: F_J completely masks F_I, denoted $F_J \vec{M} F_I$, if and only if $T(F_I|F_J) = \emptyset$, where $\emptyset$ denotes the empty set.

In words, we say that fault situation F_J completely masks F_I if and only if it is not possible to detect F_I when it is known that F_J exists in the circuit.

Friedman [7], Hayes [8], Gault *et al.* [9], and Wang [10] among others have reported on a masking relationship among faults, which we will refer to in the following as classical masking. Given $F_I, F_I \cup F_J, T(F_I)$, and $T(F_I \cup F_J)$, if for some test $X_1 \in T(F_I)$, we have that $X_1 \notin T(F_I \cup F_J)$, it is said that F_J classically masks F_I relative to the test, and if this is true for all $X \in T(F_I)$ we then say that F_J classically masks F_I. Clearly, this is not equivalent to saying that F_J completely masks F_I, since it is still possible that the fault conditions F_I and $F_I \cup F_J$ can be distinguished. Since classical masking describes a situation where tests for F_I are rendered useless by the addition of F_J to the fault condition, classical masking might be more accurately described as test set nullifying, and in the following we will denote this classical masking relationship as $F_J \vec{N} F_I$.

The concept of complete masking has also been suggested by Cha and Metze [12]. They, however, choose to refer to it as *m-covering. They then couple this m-covering concept* with test set nullifying, and refer to that resulting concept as masking.

Example 2: To illustrate the above, consider the network of Fig. 2. Let $I = \{5,6\}$ and $\alpha_I = \{\alpha_5,\alpha_6\} = \{0,1\}$, and let $J = \{13\}$ and $\alpha_J = \{\alpha_{13}\} = \{0\}$. It can be seen that $T(F_I) = \{X|x_1x_2 \oplus \overline{x_3x_4} = 1\}$ and $T(F_I \cup F_J) = \{X|x_1x_2 \oplus x_3x_4 = 1\}$. Since $T(F_I) \cap T(F_I \cup F_J) = \emptyset$, it is clear that $F_J \vec{N} F_I$. However, by Definition 1, $T(F_I|F_J) = \{X|x_1x_2\overline{x_3x_4} = 1\} \neq \emptyset$. Hence $F_J \not\vec{M} F_I$. Thus, if it were known that F_J existed in the network of Fig. 2, it would still be possible to determine whether F_I also existed. For example, if we applied the fault dependent test $X = (1100)$ to the network containing F_J, the output would be a 0 if F_I also were present, and a 1 otherwise. It should be noted that the fact that $F_J \vec{M} F_I$ does not necessarily imply that $F_I \vec{M} F_J$. Hence, although the concept of complete masking is an indication of a limit to which resolution can be achieved with algebraic terminal experiments, it should also now be clear that any such resolution testing must take into account more than just one-way masking interrelationships before definite conclusions concerning the existing fault situation within the master set can be made. The next fault interrelationship to be described moves further in that direction.

Definition 2: F_I is unresolvable from F_J, denoted $F_I \vec{R} F_J$, if and only if $T(F_I|F_J) = T(F_J|F_I) = \emptyset$.

In words, we say that fault situations F_I and F_J are unresolvable if it is not possible to detect F_I in the presence of F_J and if it is not possible to detect F_J in the presence of F_I.

The concept of unresolvability might seem to be similar to McCluskey and Clegg's concept of functional fault equivalence [11]. The differences between the two concepts are a result of the standard testing and fault detection (to which functional equivalence relates) and fault dependent testing and fault resolution (to which unresolvability relates) considerations, as will now be shown. Given the faults F_I and F_J if $Z[X;\alpha_I,Y_J] = Z[X;Y_I,\alpha_J]$ it is said that the two faults are functionally equivalent, denoted $F_I \sim F_J$. It is clear that $F_I \sim F_J$ if and only if $T(F_I) = T(F_J)$. But using the definition of unresolvability and Theorem 1, we have that $F_I \vec{R} F_J$ implies that

$$T(F_I|F_J) = T(F_J|F_I) = \emptyset$$

or

$$T(F_I) \oplus T(F_I \cup F_J) = T(F_J) \oplus T(F_I \cup F_J) = \emptyset$$

which implies

$$T(F_I) = T(F_J) = T(F_I \cup F_J).$$

Thus, the functional equivalence of two fault situations is a necessary but not sufficient condition for the unresolvability of those fault situations.

Example 3[2]: To illustrate the above, consider the circuit of Fig. 3, and let $M = \{1,2,5,6\}$, $I = \{2,6\}$, $J = \{1,5\}$, $\alpha_I = \{0,1\}$, and $\alpha_J = \{1,0\}$. Since $Z[X;\alpha_I,Y_J] = Z[X;Y_I,\alpha_J] = x_2x_3$, we have that $F_I \sim F_J$. However, note that $T(F_J|F_I) = \{X|x_2x_3 = 1\}$, and therefore, these two double faults are resolvable. In other words, if it were known that the fault F_I were in the circuit, then if we applied $X_1 = (0110)$ to the circuit, the output would be a 0 if F_J also existed and a 1 otherwise.

In the remainder of this section, we will give some useful remarks involving unresolvability, functional equivalence, complete masking, classical masking (test nullifying), and fault dependent testing. These remarks are intended to make more salient certain important features of our fault interrelationships.

Remark 1: Given the three fault situations F_I, F_J, and F_K:

a) $F_J \sim F_I \cup F_J$ if and only if $F_J \vec{R} F_I \cup F_J$.
b) $T(F_I|F_I \cup F_J) = T(F_J|F_I \cup F_J) = \emptyset$.
c) $T(F_I \cup F_J|F_K) = T(F_J|F_K) \oplus T(F_I|F_J \cup F_K)$.
d) $T(F_I|F_J) \oplus T(F_J|F_I) = T(F_I) \oplus T(F_J)$.

The proofs of these follow directly from Theorem 1 and the definitions of this section. Remark 1a) gives the conditions under which functional equivalence and unresolvability are identical interrelationships. In other words, if two fault situations are equivalent, and one of these fault situations is a subset of the other, then those two fault situations are unresolvable. Remark 1b) asserts that it is not possible to test for a subset of a specified fault situation when it is assumed that, indeed, the entire specified fault situation exists in the circuit. Remark 1c) is useful in simplifying expressions involving fault dependent tests such that all such tests are in the form of (2) wherein the assumed fault situation and the tested for situation are disjoint. For example, to determine $T(F_I \cup F_J|F_J \cup F_L)$ we can use this Remark 1c). Letting $K = J \cup L$, and noting that

$$T(F_I \cup F_J|F_J \cup F_L) = T(F_J|F_J \cup F_L) \oplus T(F_I|F_J \cup F_L) = T(F_I|F_J \cup F_L),$$

since, by Remark 1b), $T(F_J|F_J \cup F_L) = \emptyset$. Finally, Remark 1d) indicates a relationship which exists between fault dependent test sets and the composing fault test sets which does not involve the composite fault test set. This Remark 1d) makes apparent an interesting property of redundancy. Suppose, for example, that F_I and F_J are redundant fault situations. Then $T(F_I) = T(F_J) = \emptyset$. By Remark 1d) this means that $T(F_I|F_J) = T(F_J|F_I)$, and, if these two fault dependent test sets are not empty, the composing, redundant faults are resolvable in that it is possible (with the same tests) to detect either of the redundant faults in the presence of the other. This is, then, an algebraic assertion of a fault phenomena first noted by Friedman [7].

We will now consider the connection between complete masking and classical masking or test set nullifying. We know that if $F_J \vec{N} F_I$, then $T(F_I) \cap T(F_I \cup F_J) = \emptyset$. But from this it is clear that $T(F_I) \oplus T(F_I \cup F_J) \neq \emptyset$. Hence, we have that $F_J \vec{N} F_I$ implies that $F_I \not\vec{M} F_J$. In other words, if F_J classically masks F_I, then F_I does not completely mask F_J. This can be extended further by noting that if we have that $F_I \vec{N} F_J$ and $F_J \vec{N} F_I$, then clearly $F_I \vec{R} F_J$.

The above result significantly indicates the difference of the fault interrelationships which are pertinent to fault detection and fault resolution. In terms of fault detection if two fault situations classically mask each other, they form what is referred to as an undetectable fault set [9] in that the composite fault situation cannot be detected with tests from the test sets for either of the composing fault situations. However, we have seen here that in

[2] Fig. 3 and Example 3 are taken from Cha and Metze [12].

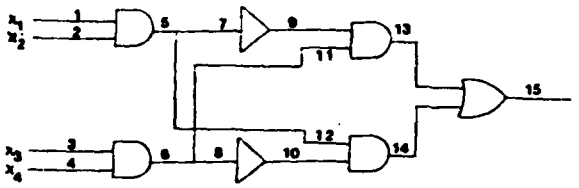


Fig. 2. $Z[X;\emptyset] = \bar{x}_1\bar{x}_2x_3x_4 + x_1x_2\bar{x}_3\bar{x}_4$

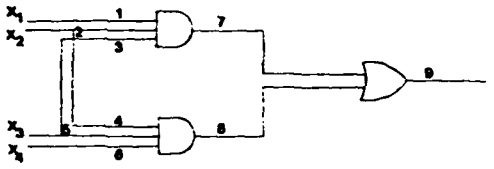


Fig. 3. $Z[X;\emptyset] = x_1x_2x_3 + x_2x_3x_4$.

terms of fault resolution, these two fault situations are then resolvable. This is summarized in the following.

Remark 2:

a) $F_I \overset{\rightarrow}{\bar{K}} F_J$ and $F_J \overset{\rightarrow}{\bar{K}} F_I$ implies $F_I \overset{\leftrightarrow}{\bar{R}} F_J$.

b) $F_I \overset{\rightarrow}{\bar{M}} F_J$ and $F_J \overset{\rightarrow}{\bar{M}} F_I$ implies $F_I \overset{\leftrightarrow}{R} F_J$.

V. Fault Interrelationship Ramifications

Whenever a test, standard or fault dependent, is applied to a network, the information contained in the outcome of that test is, in general, quite limited in terms of fault resolution. For example, if we determine, say, $T(F_I)$ for F_I, apply a $X_1 \in T(F_I)$ to the network, and observe that the network fails the test, we can only conclude that, indeed, the network is faulty. Similarly, if the network passes the test, we can only conclude that F_I is not present. Analogous statements could be made for fault dependent testing. Therefore, in order to locate or resolve the existing fault in a network, a sequence of tests must necessarily be applied. However, even with the application of a sequence of tests, a point will be reached where further resolution is not possible. We are considering in this correspondence an approach to fault resolution based on algebraic terminal experiments. For a set of master faults in a network and using this approach, the limit of resolution which can be attained is dependent upon the type of test used and the existing fault interrelationships within the master set. It should be clear that higher resolution is possible if fault dependent testing is used instead of standard testing, since fault dependent testing takes into account fault situation information not utilized by standard tests. In the following we will specify the limits of resolution attainable with fault dependent testing in terms of the fault interrelationships which were characterized in the previous section. It will be seen that in terms of resolution limits: unresolvability is to fault dependent testing as functional equivalence is to standard testing.

Suppose that in a network, we have that the existing fault in either F_I, F_J, or $F_I \cup F_J$. Now, suppose further that $F_I \overset{\leftrightarrow}{R} F_J$. We recall from the previous section that this implies that $T(F_I) = T(F_J) = T(F_I \cup F_J)$. This means, then, that if we applied standard tests for F_I, F_J, or $F_I \cup F_J$, we could not resolve the fault situation. However, suppose further that by some means it were possible to either inject F_I in the network or sense that, indeed, F_I existed in the network, or to inject F_J in the network or sense that, indeed, F_J existed in the network. Regardless, it would still not be possible even with fault dependent testing to resolve the fault situation further since $F_I \overset{\leftrightarrow}{R} F_J$. However, if we instead suppose that $F_I \sim F_J$, but where $F_I \overset{\rightarrow}{\bar{R}} F_J$, and we again had some means of externally probing or sensing conditions in the network on the lines of I or J, then in this case it may be possible, in general, to further resolve the existing fault situation since the implication of functional equivalence is simply that $T(F_I) = T(F_J) \neq T(F_I \cup F_J)$. Hence, unresolvability can be considered to be an interrelationship among faults which limits the degree of resolution which can be attained when testing is performed with more information regarding the existing fault situation than is generally used with standard test set approaches. Hence, the parallel between functional equivalence/standard testing and unresolvability/fault dependent testing is clear.

It might be noted here that, indeed, if we did have some external means of injecting faults into the network on, say, some lines K of the circuit, and we had the case that $F_I \overset{\leftrightarrow}{R} F_J$, but that $F_F \vec{M} F_J$, and $F_K \overset{\rightarrow}{\bar{R}} F_I$, then we could resolve the fault situation further. As an illustration of this, consider the following.

Example 4: It can be observed in the network of Fig. 1 that if $I = \{5\}$ and $\alpha_I = \{\alpha_5\} = \{1\}$, and $J = \{9\}$ and $\alpha_J = \{\alpha_9\} = \{1\}$, then $F_I \overset{\leftrightarrow}{R} F_J$. Thus, even if it were possible to inject one of these two faults, say, F_J, it is easily seen here that it still would not be possible to determine whether F_I also existed in the network. However, if we could inject some other fault, say, F_K, $K = \{13\}$, $\alpha_K = \{\alpha_{13}\}$, $= \{0\}$, then since $F_K \overset{\rightarrow}{\bar{M}} F_J$, and $F_K \overset{\rightarrow}{\bar{M}} F_I$, applying the input combination $(1111) \in T(F_I|F_K)$ we can now test for the fault F_I.

A complete treatment of this idea of enhancing resolvability by means of external lines to the network which can be used to inject fault situations is beyond the scope of this paper. However, it seems that the functional approach to fault interrelationships being utilized in this correspondence makes clear the criteria which must be satisfied for the design of highly resolvable networks.

VI. Conclusion

In conclusion, we have presented in this correspondence a characterization of fault interrelationships which limit the resolution of a fault situation from within a master set of faults when we have certain information about that fault situation which allows us to do fault dependent testing instead of standard testing. This has been done by means of an extension of the Boolean difference to fault dependent situations. It should be clear that unresolvability, the main interrelationship of this characterization, is really only another point on a spectrum of fault interrelationships which is increasingly being seen to be of significance in fault analysis. Certainly, this spectrum will be worthy of future research.

Acknowledgment

The authors wish to thank Prof. G. Metze of the Department of Electrical Engineering and the Coordinated Sciences Laboratory of the University of Illinois at Urbana-Champaign for suggestions, criticism, and discussion regarding this work.

References

[1] C. T. Ku and G. M. Masson, "The Boolean difference and multiple fault analysis," *IEEE Trans. Comput.*, vol. C-24, pp. 62–71, Jan. 1975.

[2] A. Thayse and M. Davis, "Boolean differential calculus and its application to switching theory," *IEEE Trans. Comput.*, vol. C-22, pp. 409–420, Apr. 1973.

[3] S. S. Yau and Y. S. Tang, "An efficient algorithm for generating complete test sets for combinational logic circuits," *IEEE Trans. Comput.*, vol. C-20, pp. 1245–1251, Nov. 1971.

[4] F. F. Sellers, M. Y. Hsiao, and L. W. Bearnson, "Analyzing errors with the Boolean difference," *IEEE Trans. Comput.*, vol. C-17, pp. 676–683, July 1968.

[5] A. C. L. Chiang, I. S. Reed, and A. V. Banes, "Path sensitization, partial Boolean difference, and automated fault diagnosis," *IEEE Trans. Comput.*, vol. C-21, pp. 189–195, Feb. 1972.

[6] S. B. Akers, Jr., "On the theory of Boolean functions," *SIAM J. Appl. Math.*, vol. 7, pp. 487–498, 1959.

[7] A. D. Friedman, "Fault detection in redundant circuits," *IEEE Trans. Electron. Comput.*, vol. EC-16, pp. 99–100, Jan. 1967.

An Efficient Fault Diagnosis Algorithm for Symmetric Multiple Processor Architectures

GERARD G. L. MEYER AND GERALD M. MASSON

Abstract— A new diagnosis algorithm for determining the existing fault situation in a symmetric multiple processor architecture is given. The algorithm assumes that there are n processors, each of

Manuscript received January 5, 1977; revised May 17, 1977 and August 29, 1977. This work was supported by the Office of Naval Research under Contract NR 048 630.

The authors are with the Department of Electrical Engineering, The Johns Hopkins University, Baltimore, MD 21218.

0018-9340/78/1100-1059$00.75

which is tested by at least t other processors, and at most t of which are faulty. The existing fault situation is always diagnosed if $n \geq 2t + 1$ and, in some cases, can still be diagnosed if $n < 2t + 1$. The implementation of the algorithm is straightforward and suitable for microprocessor applications.

***Index Terms*—Diagnosis algorithm, fault syndromes, microprocessor, modular networks.**

Introduction

Consider a general model of a multiple processor architecture consisting of n digital modules denoted $U_0, U_1, \cdots, U_{n-1}$ and some associated interconnection design denoted $D_{\delta,t}$. These modules, for example, could be n processors implementing a segmented algorithm [6]. Regardless of the use of the multiple processor architecture, we will assume that each U_i is capable of testing the other U_j's to which it is directly connected for some specified class of faults. If a module contains any such fault, we will refer to it as faulty. The problem we will study in this paper is the diagnosis of an existing fault situation among the modules given their respective testing results. This problem is not new and has been examined elsewhere in the literature [1], [3], [4], [5], [7], [8]. The results to be presented here represent a new approach to such diagnosis. In particular, the diagnosis procedure described will be seen to be sufficiently straightforward to be easily implementable on a simple processor, e.g., a microprocessor, and for a proper interconnection design among the processors and upper bound on the number of simultaneous faults which can occur, will always yield the correct diagnosis of the existing fault situation.

Preliminaries

Given n modules $U_0, U_1, \cdots, U_{n-1}$, we will denote the modules which U_i tests by $U_{f(r,i)}$, $r = 1, 2, \cdots, t$ where $f(r, i) \in [0, 1, \cdots, n-1]$, $i = 0, 1, \cdots, n-1$. For convenience, we will always assume that U_i tests itself and, regardless of its state, concludes that it is fault free. The outcome of the test of module $U_{f(r,i)}$ by module U_i will be denoted $a(i, f(r, i))$ where

$$a(i, f(r, i)) = \begin{cases} 0, & \text{if } U_i \text{ concludes that } U_{f(r,i)} \text{ is fault free} \\ 1, & \text{otherwise.} \end{cases}$$

It should be noted that the conclusion of, say, U_i regarding the state (faulty or fault free) of the modules to which it is connected is only reliable if indeed U_i is fault free. If with each module U_i we associate a test table B_i, $i = 0, 1, \cdots, n-1$, where B_i represents the conclusion of U_i regarding the states of all the modules, we have the problem of determining the existing fault situation based on the available test results. Whether or not this is feasible clearly depends on the number of faults and the interconnection design. We will assume in the following that at most t modules can be simultaneously faulty and that every module is tested by at least t other modules. Under some assumptions on the interconnection design, Preparata, Metze, and Chien [7] have shown that it is feasible to diagnose any valid fault situation. However, the diagnosis algorithms which have been proposed to do so are quite complex [1], [4], [5]. We propose here a new diagnosis algorithm for this problem. For the purpose of explanation we will assume in the following that the interconnection design between the modules is the so-called $D_{1,t}$ design of [7], wherein there is a testing interconnection from U_i to U_j if and only if $j - i = m$ (modulo n) and m assumes the values from 1 to t. The results presented here have been extended to more general interconnection designs, but since they are descriptively cumbersome, these extensions will not be detailed.

Diagnosis Algorithm

Each test table B_i has components $B_{i,0}, B_{i,1}, \cdots, B_{i,n-1}$ where $B_{i,j}$ represents the conclusion of module U_i regarding the state of module U_j. If module U_i "believes" that module U_j is fault-free, then $B_{i,j}$ is set to the value 0; otherwise, $B_{i,j}$ is set to the value 1. Suppose that $B_0, B_1, \cdots, B_{n-1}$ are complete in the sense that every module has a conclusion regarding the state of each of the modules U_i, $i = 0, 1, \cdots, n-1$. We will assume here that if a module is fault-free, its corresponding table is correct.

Lemma 1: There exists at least $n - t$ of the B_i tables which are identical.

Proof: Since at most t modules are faulty, and since a table corresponding to a fault-free module correctly describes the fault situation, the theorem follows.

Lemma 2: If there exists only one set of identical tables $B_{i(1)}, B_{i(2)}, \cdots, B_{i(s)}$, such that $s \geq n - t$, then each of these tables in this set correctly describes the existing fault situation.

Proof: We already know that there exists at least $n - t$ correct, and therefore identical, tables. Therefore, if only one set of identical tables has a cardinality larger than or equal to $n - t$, this set must consist of the correct tables.

It should be clear that no conclusion can be made regarding the fault situation if there exists more than one set of identical tables with cardinality larger than or equal to $n - t$.

Theorem 1: Suppose that $n \geq 2t + 1$; then there exists one and only one set of identical tables with cardinality larger than or equal to $n - t$.

Proof: Suppose that $n \geq 2t + 1$ and assume that there exist two sets of identical tables of cardinality n_1 and n_2, respectively. Assume

$$n_1 \geq n - t$$

and

$$n_2 \geq n - t.$$

Then

$$n_1 + n_2 \geq 2n - 2t.$$

We know that

$$n \geq n_1 + n_2$$

and therefore

$$n \geq 2n - 2t.$$

This inequality, used in conjunction with $n \geq 2t + 1$, yields

$$2n \geq 2n + 1$$

and we conclude that we cannot have two sets of identical tables of cardinality larger than or equal to $n - t$ when $n \geq 2t + 1$.

At this point we need an efficient procedure to build the complete n tables $B_0, B_1, \cdots, B_{n-1}$ such that if module U_i is fault-free, then the table B_i reflects accurately the fault situation of the multiple processor architecture. Such an algorithm is presented in the following to compute the tables $B_0, B_1, \cdots, B_{n-1}$.

Algorithm 1: Let i in $[0, 1, \cdots, n-1]$ and t in $[1, 2, \cdots, n-1]$ be given.

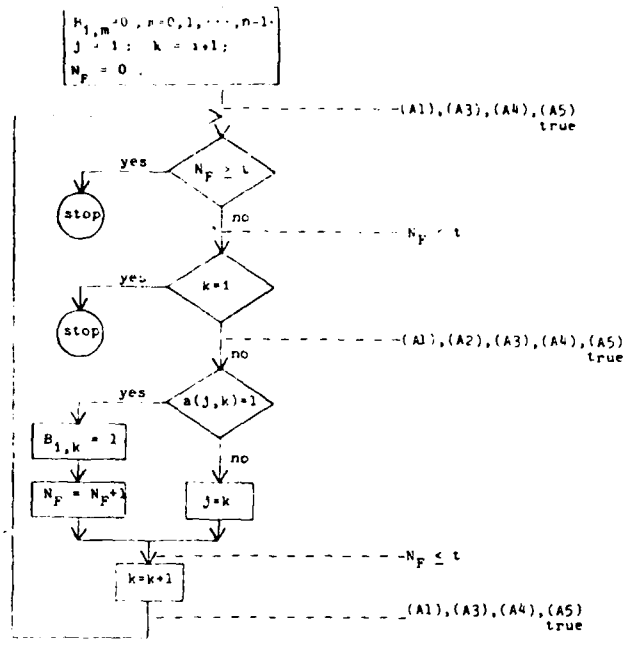


Fig. 1. Flowchart of Algorithm 1. Interpretation when module U_i is not faulty.

Step 0: Set $B_{i,m}=0$ for $m=0, 1, \cdots, n-1$, set $j=i$, set $k=i+1$, and set $N_F=0$.

Step 1: If $N_F \geq t$, stop; else, go to Step 2.

Step 2: If $k=i$, stop; else, go to Step 3.

Step 3: If $a(j,k)=1$, set $B_{i,k}=1$, set $N_F=N_F+1$, and go to Step 4; else, set $j=k$ and go to Step 4.

Step 4: Set $k=k+1$ and go to Step 1.

Notes:

1) All additions are performed modulo n.

2) We assume a $D_{1,t}$ interconnection design, i.e., $f(r,j)=j+r$ (modulo n), and therefore, we use the notation $a(j, j+r)$ instead of the more general notation $a(j, f(r,j))$.

Theorem 2: If a $D_{1,t}$ interconnection design is used, if the maximum number of faults which may occur is t, and if module U_i is fault-free, then the table B_i constructed by the algorithm accurately reflects the existing fault situation.

Proof: We need to show that the algorithm is well defined and that it produces tables B_i which are correct whenever U_i is not faulty. The technique we use to prove the theorem is based on the use of invariant assertions as described in [2] (see Fig. 1).

We assume that a $D_{1,t}$ interconnection design is used, i.e., module U_i tests the modules $U_{i+1}, U_{i+2}, \cdots, U_{i+t}$. The algorithm uses the quantity $a(j,k)$ which contains the result of the test of module k by module j. It follows that the algorithm is well defined if and only if j and k are related by

$$k=j+r$$

where r is some integer in $[1, 2, \cdots, t]$.

Assume that before executing Step 3, the following assertion holds:

$$\text{(A1)}\quad j+1 \leq k \leq j+1+N_F.$$

Then it can be shown that (A1) still holds after the execution of Step 4. Clearly, (A1) is satisfied by the initial values given to j, k, and N_F, and therefore we conclude that (A1) is always satisfied before the execution of Step 3.

It is only possible to reach Step 3 if

$$N_F < t.$$

It follows that just before the execution of Step 3, the quantities j and k are related by the assertion

$$\text{(A2)}\quad j+1 \leq k \leq j+t$$

which shows that the algorithm is well defined.

The first part of the proof showed that the algorithm is well defined. We now prove that if U_i is fault-free, then the table B_i reflects the actual fault situation. Following again the approach described in [2], we show that the following assertions are always satisfied before the execution of Step 3.

(A3) The module U_j is not faulty.

(A4) B_i accurately reflects the existing fault situation up to $k-1$, i.e., for all m in $[i, i+1, i+2, \cdots, k-1]$:

$$B_{i,m}=0 \quad \text{if and only if module } U_m \text{ is not faulty}$$

and

$$B_{i,m}=1 \quad \text{if and only if module } U_m \text{ is faulty.}$$

(A5) N_F contains the number of faulty modules up to $k-1$, i.e.,

$$N_F=\sum_{m=i}^{k-1} B_{i,m}.$$

It can be shown that if (A3), (A4), and (A5) are true before the execution of Step 3, then they are still true after the execution of Step 4. Clearly, (A3), (A4), and (A5) hold after the execution of Step 0, and therefore we conclude that (A3), (A4), and (A5) are always true before the execution of Step 3.

Now, suppose that the algorithm stops in Step 1. We know that B_i is correct up to k and that $N_F=t$. In other words, $B_{i,m}$ correctly

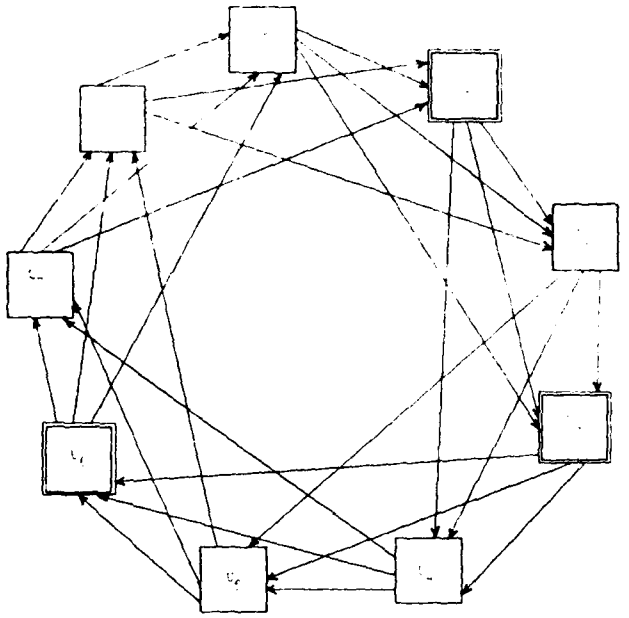


Fig. 2. $D_{1,3}$ interconnection of nine modules; modules U_1, U_3, and U_6 faulty.

reflects the fault situation for $m = i, i+1, \cdots, k$ and t faults have been detected. But we have assumed that at most t faults may occur, and therefore this implies that the remaining modules are not faulty. The $B_{i,m}$ for $m = k+1, k+2, \cdots, i-1$ are equal to 0, and therefore the complete table B_i is correct.

Suppose that the algorithm instead stops in Step 2; then B_i is correct up to $k = i$, and therefore B_i is correct.

Although we have shown that when the algorithm stops, it produces the correct table, it remains to be shown that it indeed stops after a finite number of iterations. We note that k takes the values $i, i+1, i+2, \cdots$, and therefore if the algorithm does not stop in Step 1, it must necessarily stop in Step 2. This concludes the proof of the theorem.

Note that table B_i does not contain any DON'T CARE entries. This occurs because table B_i is constructed by using not only the test results of module U_i, but also the test results of other modules. For example, if U_i does not test directly U_j, but U_i tests U_k and finds U_k correct, then U_i accepts the conclusion of U_k regarding U_j.

Accelerated Algorithm

The diagnosis of the set of faulty modules based on the results of Lemmas 1 and 2 and Theorem 1 requires that the table B_i, $i = 0, 1, \cdots, n-1$ be compared. This process is time consuming and may be avoided. For each $j = 0, 1, \cdots, n-1$, let γ_j be the number of indices i for which $B_{i,j} = 1$, i.e.,

$$\gamma_j = \text{cardinality of } \{i \in [0, 1, \cdots, n-1] \mid B_{i,j} = 1\},$$

then these quantities may be used in a diagnostic algorithm as follows.

Algorithm 2: Let t in $[1, n-1]$ be given.

Step 0: Compute the tables $B_0, B_1, \cdots, B_{n-1}$ by using Algorithm 1.

Step 1: Compute the quantities $\gamma_0, \gamma_1, \cdots, \gamma_{n-1}$.

Step 2: Let $V = \{j \in [0, 1, \cdots, n-1] \mid \gamma_j \geq t+1\}$.

Note that if the tables B_i are arranged to form an $n \times n$ array where B_i is the ith row, then γ_j is just the number of 1's in column j.

Theorem 3: If a $D_{1,t}$ interconnection design is used, if the maximum number of faults which may occur is t, and if $n \geq 2t+1$, then U_j is faulty if and only if j is in V.

Proof: The result is a direct consequence of Lemmas 1 and 2 and Theorems 1 and 2.

Algorithm 2 is well suited for implementation on a microprocessor. For example, on an Intel 8080 microprocessor, the total amount of memory necessary to store the data and the program in the case $n = 8$ and $t = 2$ is 176 words of 8 bits, i.e., 1408 bits.

We note that Algorithm 2 may be implemented in parallel on a network of N microprocessors with $N \leq n$. In particular, if N microprocessors are used, then it is possible to compute in parallel all the tables B_i and all the quantities γ_j. The computational time necessary to diagnose the network of n modules using N microprocessors for implementing Algorithm 2 is essentially $T\lceil n/N \rceil / n$ where T is the computational time necessary to execute the instructions of Algorithm 2 when a single microprocessor is used and $\lceil n/N \rceil$ is the smallest integer larger than n/N. The simplicity of Algorithm 2 gives a large amount of flexibility to its implementation. For example, one could construct a special very reliable arbitrator device which would receive all the $a(i, j)$'s from the modules and then would generate the diagnosis of the system using Algorithm 2. The high reliability of the arbitrator could be justified because of the simplicity of its required operation relative to the complexity of the modules in the network. Alternatively, it would be possible to send the $a(i, j)$'s to each module, which could in turn compute the tables B_i. In this case, each module could then decide for itself which other modules are faulty and act accordingly. Clearly, many other variations for implementation are possible.

1			0	1	1				
2				1	0	0			
3					0	0	1		
4						0	1	0	
5							1	0	0
6	1							1	1
7	0	1							0
8	0	1	0						

Fig 3. Test outcomes $a(j, k)$: modules U_1, U_3, and U_6 faulty

i \ k	0	1	2	3	4	5	6	7	8
0	0	1	0	1	0	0	1	0	0
1	0	0	0	1	0	0	1	0	0
2	0	1	0	1	0	0	1	0	0
3	0	1	0	0	0	0	1	0	0
4	0	1	0	1	0	0	1	0	0
5	0	1	0	1	0	0	1	0	0
6	0	0	0	0	0	0	0	1	1
7	0	1	0	1	0	0	1	0	0
8	0	1	0	1	0	0	1	0	0

Fig 4. Quantity $B_{i,j}$ obtained by using the algorithm to process the test results given in Fig 3

Example

In order to demonstrate the simplicity of the algorithms, we apply them to the network given in Fig. 2. The network contains $n = 9$ modules, $t = 3$, i.e., at most three modules may be faulty, and a $D_{1,3}$ interconnection design is used, i.e., module U_0 tests U_1, U_2, and U_3, module U_1 tests U_2, U_3, and U_4, etc. Assume that the modules U_1, U_3, and U_6 are faulty. Fig. 3 contains a possible set of test outcomes. The application of the algorithm to these test outcomes yields the tables B_i, $i = 0, 1, 2, \cdots, 8$ given in Fig. 4. We find that the tables B_0, B_2, B_4, B_5, B_7, and B_8 are identical. We have six identical tables, and using Lemma 2, we conclude that these tables reflect the correct fault situation of the network, i.e., we conclude that the modules U_1, U_3, and U_6 are faulty. Alternatively, we may compute the quantities γ_j, i.e., $\gamma_0 = 0$, $\gamma_1 = 8$, $\gamma_2 = 0$, $\gamma_3 = 8$, $\gamma_4 = 0$, $\gamma_5 = 0$, $\gamma_6 = 8$, $\gamma_7 = 1$, and $\gamma_8 = 1$, and then compute the set $V = \{j | \gamma_j \geq 4\} = \{1, 3, 6\}$. Using Theorem 3, we conclude once again that U_1, U_3, and U_6 are faulty.

Conclusion

An approach to the problem of fault diagnosis of symmetric multiple processor architectures has been proposed. It consists of constructing tables B_i, assuming that the corresponding modules U_i are not faulty, followed by a voting procedure. The construction of the tables B_i is decoupled in the sense that each table may be constructed independently of the others. It is possible to decrease the amount of computation necessary to obtain all the tables B_i, $i = 0, 1, \cdots, n - 1$ by increasing the dependency between the construction of the various tables. It is not difficult to find schemes in which the construction of the table U_j depends on the tables $U_0, U_1, \cdots, U_{j-1}$. Such schemes are more complicated to code than the one we propose, require more memory to store the program, and do not lend themselves to parallel implementation. Therefore, we feel that our scheme, i.e., Algorithm 2 which has a time complexity of $0(n^2)$ if sequentially implemented and $0(n)$ if implemented on a network of n microprocessor, is ideally suited for the fault diagnostic of $D_{1,t}$ networks.

References

[1] A. M. Corluhan and S. L. Hakimi, "On an algorithm for identifying faults in a T-diagnosable system," in *Proc. 1976 Conf. on Inform. Sci. and Syst.*, The Johns Hopkins Univ., 1976, pp. 370-375.

[2] R. W. Floyd, "Assigning meanings to programs," in *Proc. Symp. in Applied Math.*, Amer. Math. Soc., Providence, RI, 1967, pp. 19-32.

[3] S. L. Hakimi and A. T. Amin, "Characterization of the connection assignment of diagnosable systems," *IEEE Trans. Comput.*, vol. C-23, pp. 86-88, Jan. 1974.

[4] T. Kameda, S. Toida, and F. Allan, "A diagnosing algorithm for networks," *Inform. Contr.*, vol. 29, pp. 141-148, 1975.

[5] S. N. Maheshwari and S. L. Hakimi, "On models of diagnosable systems and probabilistic fault diagnosis," *IEEE Trans. Comput.*, vol. C-25, pp. 228-236, Mar. 1976.

[6] G. G. L. Meyer, "A segmented algorithm for solving a class of constrained discrete optimal control problems," *IEEE Trans. Automat. Contr.*, vol. AC-19, pp. 134-136, Apr. 1974.

[7] F. P. Preparata, G. Metze, and R. T. Chien, "On the connection assignment problem of diagnosable systems," *IEEE Trans. Electron. Comput.*, vol. EC-16, pp. 848-854, Dec. 1967.

[8] J. D. Russell and C. R. Kime, "On the diagnosability of digital systems," in *Proc. 1973 Int. Symp. on Fault Tolerant Computing*, IEEE Computer Society Publications, June 1973, pp. 139-144.

Recursive Coverage Projection of Test Sets

VINOD K. AGARWAL AND GERALD M. MASSON

Abstract—In the generation of test sets for the detection of stuck-type faults in combinational switching networks, it is an expedient and reasonably common assumption to consider explicitly faults only of specified sizes (for example, all single faults), and then to assume (or hope) that most or all faults of larger sizes will be covered (that is, detected) as well. This paper systematically addresses this aspect of multiple fault coverage in a quantitative manner for combinational networks, wherein only primary input fanout is allowed. A procedure is given to estimate (or project) the multiple fault coverage capability of a test set based on the known coverage capability of that test set for subsets of the multiple faults. This is accomplished by means of a recursive use of a detailed formula which exploits two fundamental interrelationships between test sets and faults. Based upon these results, it can be shown that the above-mentioned assumption must be made, in general, with discretion as its validity is highly network structure/test set dependent.

Index Terms—Consistency, coverage, fault detection, internal fanout-free networks, multiple faults, recursive projection.

I. Introduction

The magnitude of the number of multiple faults which can potentially occur in any logic network of reasonable size usually prohibits their individual consideration in test set generation. Nevertheless, the ability to test a logic network for multiple faults with a certain degree of confidence is fundamental to the reliability aspects of a wide range of systems composed of such logic networks [9]–[12]. Thus, in practice, it is often found that a test set will be generated explicitly for only a subset of all possible faults which can occur with the assumption (or hope) that most other faults are covered as well. Except, however, for some reasonably straightforward results, there has heretofore been little work pub-

Manuscript received March 15, 1979; revised May 9, 1979. This work was supported by the Office of Naval Research under Contract N00014-75-C-1196.

V. K. Agarwal is with the Department of Electrical Engineering, McGill University, Montreal, P.Q., Canada.

G. M. Masson is with the Department of Electrical Engineering, The Johns Hopkins University, Baltimore, MD 21218.

0018-9340/79/1100-0865$00.75

lished in the literature which theoretically analyzes this assumption in a quantitative manner. This note, therefore, addresses this problem of analyzing the coverage capability of test sets for multiple faults of sizes larger than that for which the test sets were explicitly generated.

A particularly prominent aspect of this problem is the case in which a complete single fault test set is generated for a network and then used for the coverage of both single and multiple faults in the network. The following example illustrates the tenuous nature of such an approach.

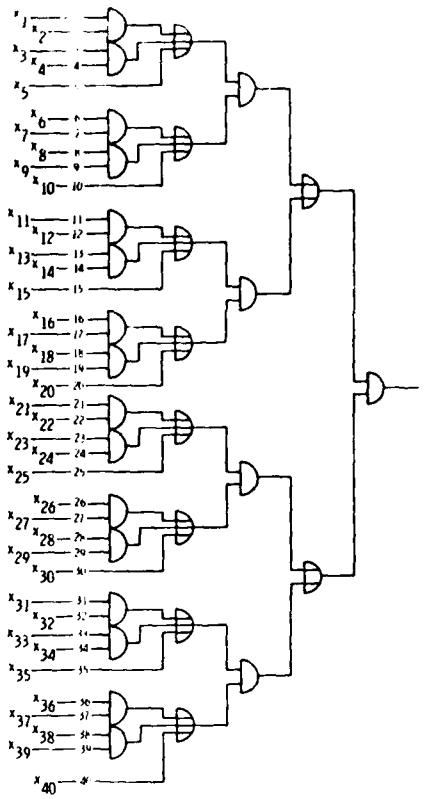
Example 1: Consider the 40-input, single-output, nonredundant, combinational network, denoted as N, shown in Fig. 1(a). Let $K = \{1, 3, 6, 8, 11, 13, 16, 18, 21, 23, 26, 28, 31, 33, 36, 38\}$ be a set of $v = 16$ lines in N, and let T_s be the single fault detection test set (SFDTS) consisting of the 56 input vectors shown in Fig. 1(b). Now it can be seen upon investigation of this network that T_s also covers every multiple fault involving any 2 and 3 of these 16 lines (indeed, this is expected [2], [4]) and approximately 90 percent of the multiple faults involving any 4, 5, and 6 of these 16 lines. However, if we consider the multiple faults simultaneously involving all 16 lines, then it can be seen (on the basis of coverage projection results to be later developed in this paper) that of the $2^{16} = 65\,536$ such multiple faults, only 29 822 of these faults are covered by T_s. That is, more than one-half of all the multiple faults involving all 16 lines of K can be shown to be undetected by the given T_s. (It is interesting to note, incidentally, that no two or more of these 2^{16} multiple faults are functionally equivalent.) Thus, we see that for this network and these 16 lines, the multiple fault coverage capability for the given T_s is reasonable for multiple faults involving up to 6 lines, but that for the multiple faults of greater size, this capability is quite tenuous.

The above example is not an isolated or rare situation as the perspective given by the results to be developed in this paper will indicate. Accordingly, we are well motivated here to develop a formal, systematic approach for evaluating the multiple fault coverage capability of test sets.

Finally, the fact that we deal in this paper exclusively with networks wherein only primary input fanout is allowed should not be viewed as overly restrictive. While it would be inappropriate here to detail the importance of their position in the general area of combinational switching networks, it is clear that such networks have provided and will continue to provide a useful framework for the establishment of general results and guidelines in the field of fault analysis. Moreover, since the use of our results leads to somewhat of a warning regarding the projected multiple fault coverage capability of test sets, these networks serve us well, as it has been shown [6], [7] that the presence of internal fanout should, if anything, decrease this coverage capability. It should be noted that the class of internal fanout-free networks considered in this paper is significantly more general than the class of tree networks wherein no fanout is allowed. In fact, straightforward methods to determine complete multiple fault detection test sets for tree networks are well known [8]–[12].

II. Preliminaries

An n-input, single-output internal fanout-free combinational switching network in which primary input fanout is allowed will be denoted in the following simply as N. For reference, each line of N will be labeled with a distinct integer. Subsets of the set of lines in N will then be denoted, for example, as $I = \{i_1, \cdots, i_p\}$, $J = \{j_1, \cdots, j_q\}$, or $K = \{k_1, \cdots, k_v\}$. The n primary input variables $x_1, x_2, \cdots, x_n$ to N will be collectively denoted in general by an n-tuple input vector X. We will use X_i to signify a particular input vector to the network wherein each $x_j \in X_i$, $1 \le j \le n$, is assigned a specific binary value of either 0 or 1.



(a)

i	Input Vector X_i^*	i	Input Vector X_i^*
1	$(3, 5, 11, 13, 15)_0$	29	$(12, 18, 19, 31, 32, 38, 39)_1$
2	$(1, 5, 11, 13, 15)_0$	30	$(14, 16, 17, 33, 34, 36, 37)_1$
3	$(8, 10, 16, 18, 20)_0$	31	$(13, 14, 17, 33, 34, 36, 37)_1$
4	$(6, 10, 16, 18, 20)_0$	32	$(11, 12, 19, 31, 32, 38, 39)_1$
5	$(1, 3, 5, 13, 15)_0$	33	$(1, 8, 9, 21, 22, 28, 29)_1$
6	$(1, 3, 5, 11, 15)_0$	34	$(3, 6, 7, 23, 24, 26, 27)_1$
7	$(6, 8, 10, 18, 20)_0$	35	$(3, 4, 6, 23, 24, 26, 27)_1$
8	$(6, 8, 10, 16, 20)_0$	36	$(1, 2, 8, 21, 22, 28, 29)_1$
9	$(1, 2, 3, 4, 11, 13, 15)_0$	37	$(11, 18, 19, 31, 32, 38, 39)_1$
10	$(6, 7, 8, 9, 16, 18, 20)_0$	38	$(13, 16, 17, 33, 34, 36, 37)_1$
11	$(1, 3, 5, 11, 12, 13, 14)_0$	39	$(13, 14, 16, 33, 34, 36, 37)_1$
12	$(6, 8, 10, 16, 17, 18, 19)_0$	40	$(11, 12, 18, 31, 32, 38, 39)_1$
13	$(23, 25, 31, 33, 35)_0$	41	$(1, 2, 8, 9, 22, 28, 29)_1$
14	$(21, 25, 31, 33, 35)_0$	42	$(3, 4, 6, 7, 24, 26, 27)_1$
15	$(28, 30, 36, 38, 40)_0$	43	$(3, 4, 6, 7, 23, 24, 27)_1$
16	$(26, 30, 36, 38, 40)_0$	44	$(1, 2, 8, 9, 21, 22, 29)_1$
17	$(21, 23, 25, 33, 35)_0$	45	$(11, 12, 18, 19, 32, 38, 39)_1$
18	$(21, 23, 25, 31, 35)_0$	46	$(13, 14, 16, 17, 34, 36, 37)_1$
19	$(26, 28, 30, 38, 40)_0$	47	$(13, 14, 16, 17, 33, 34, 37)_1$
20	$(26, 28, 30, 36, 40)_0$	48	$(11, 12, 18, 19, 31, 32, 39)_1$
21	$(21, 22, 23, 24, 31, 33, 35)_0$	49	$(1, 2, 8, 9, 21, 38, 29)_1$
22	$(26, 27, 28, 29, 36, 38, 40)_0$	50	$(3, 4, 6, 7, 23, 26, 27)_1$
23	$(21, 23, 25, 31, 32, 33, 34)_0$	51	$(3, 4, 6, 7, 23, 24, 26)_1$
24	$(26, 28, 30, 36, 37, 38, 39)_0$	52	$(1, 2, 8, 9, 21, 22, 28)_1$
25	$(2, 8, 9, 21, 22, 28, 29)_1$	53	$(11, 12, 18, 19, 31, 38, 29)_1$
26	$(4, 6, 7, 23, 24, 26, 27)_1$	54	$(13, 14, 16, 17, 33, 36, 37)_1$
27	$(3, 4, 7, 23, 24, 26, 27)_1$	55	$(13, 14, 26, 17, 33, 34, 36)_1$
28	$(1, 2, 9, 21, 22, 28, 29)_1$	56	$(11, 12, 18, 19, 31, 32, 38)_1$

(b)

*$X_i = (e_1, e_2, \cdots, e_n)_\beta$, $\beta \in \{0, 1\}$ implies that the 40-tuple input vector X_i is such that the components at the positions $e_1, e_2, \cdots, e_n$ are all β, and the components at all the other remaining positions are β. For example, $X_{25} = (0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)$.

Fig. 1. (a) Network for Example 1. (b) Test Set for Example 1.

The Boolean function in terms of the primary input variables realized by any line, say, t in N, will be denoted by Y_t. We will refer to Y_t as the internal line variable for line t. Similarly, we can succinctly describe the set of Boolean functions corresponding to the set of internal line variables of, say, the set of lines $K = \{k_1, k_2,$

$\cdots, k_v\}$ as $Y_K = \{Y_{k_1}, Y_{k_2}, \cdots, Y_{k_v}\}$. The primary output function of N can, of course, be expressed entirely in terms of the primary input variables. However, for our purposes, it will be necessary to also express the primary output function in terms of internal line variables. Accordingly, we let $Z[X; Y_K]$ denote the primary output function of N when it is written in terms of X and Y_K. For example, if $K = \{1, 2, 3, 4\}$, the Boolean function $Z[X; Y_K]$ for a network N might be written as $Z[X; Y_K] = (x_1\bar{x}_2 Y_1 + x_3)\cdot(x_4 \bar{Y}_2) + x_1x_3 \bar{Y}_3 + Y_4 + x_2x_5$.

Given any set K of v lines in N, clearly, there are 2^v possible stuck-type fault events that can be associated with these v lines wherein each of the lines is either stuck-at-zero or stuck-at-one. This set of 2^v such fault events on the set K of lines in N will be referred to as the fault complex on K and will be denoted as $F_K = \{\alpha_K^1, \alpha_K^2, \cdots, \alpha_K^{2^v}\}$ where each $\alpha_K^i \in F_K$ is described by a binary v-tuple. In the presence of any fault event $\alpha_K^i \in F_K$ in N, the resulting faulty primary output function of N can be written as $Z[X; \alpha_K^i]$. Clearly, this is obtained from $Z[X; Y_K]$ by simply replacing the variables of Y_K in the function by their corresponding binary values from α_K^i. In the following, whenever there are no chances of ambiguity, α_K will be used in place of α_K^i to denote a general element of F_K.

It should be noted here that because we are considering only internal fanout-free networks where primary input fanout is allowed, it follows that there are dominating interrelationships among the faults on the lines of N which indicate that not all subsets of lines of N are really of interest. Accordingly, we will make the following two assumptions on the members of sets of lines to be considered in this paper: first, we will assume that any set of lines we consider, such as the set K discussed above, will not contain primary input fanout stem lines; second, we will assume that any set of lines we consider will contain no two or more lines which are on the same path from a primary input to the primary output. It should be clear that these assumptions in no way restrict the consideration of fault events of interest. It should also be noted that because of these assumptions, it follows that the primary output function $Z[X; Y_K]$ for any set K of lines of interest in N can be expressed such that each internal line variable for each $t \in K$ occurs exactly once in the function, either as Y_t or as $\bar{Y}_t$. Finally, it should be clear that because of this latter property of $Z[X; Y_K]$ regarding the single occurrence of each $Y_t \in Y_K$, for any set K of lines in N, we have that $Z[X; Y_K]$ can be written as a function of Boolean subfunctions such that none of the Y_t variables appears in more than one of the subfunctions.

More particularly, let K be a set of $v \geq 2$ lines in N, and let I and J be disjoint, nonempty subsets of K such that $I \cup J = K$. Consider then the following.

Definition 2.1: The sets I and J are said to form a 2-partition of K if there exist two Boolean subfunctions, say, $H[X; Y_I]$ and $M[X; Y_J]$, such that the primary output function of N can be written as $Z[X; Y_K] = H[X; Y_I] * M[X; Y_J] + 0[X]$ where "$*$" corresponds to either "$+$" or "$\cdot$".

Example: If $Z[X; Y_K] = (x_1\bar{x}_2 Y_1 + x_3)\cdot(x_4 \bar{Y}_2) + x_1x_3 \bar{Y}_3 + Y_4 + x_2x_5$, then $I = \{1, 2\}$ and $J = \{3, 4\}$ form a 2-partition of $K = \{1, 2, 3, 4\}$, wherein $H[X; Y_I] = (x_1\bar{x}_2 Y_1 + x_3)\cdot(x_4 Y_2)$, $M[X; Y_J] = x_1x_3 \bar{Y}_3 + Y_4$, and $0[X] = x_2x_5$.

Because of the above-mentioned assumptions on the members of sets of lines considered in this paper, it should be clear that given any set K of $v \geq 2$ lines in N, there will always exist at least one 2-partition of K. This concept of a 2-partition will play an important role in our coverage projection theory. Thus, it will be convenient in the following to refer to each of the sets I and J as component sets of K, and to refer to each of the fault complexes F_I and F_J as component fault complexes of F_K. Note that if the component sets I, say, $I = \{i_1, i_2, \cdots, i_p\}$, and J, say, $J = \{j_1, j_2, \cdots, j_q\}$, are considered as ordered vectors, such as $(i_1, \cdots, i_p)$ and $(j_1, \cdots, j_q)$, respectively, and if the set K is accordingly ordered as $(i_1, i_2, \cdots, i_p, j_1, j_2, \cdots, j_q)$, then the fault complex F_K will be isomorphic to the Cartesian product $F_I \times F_J$. Throughout this paper, the above-mentioned ordering in sets I, J, and K will be assumed for the sake of notational convenience. Also, we will write that $F_K = F_I \times F_J$ where it will be understood that elements of F_K are v-tuples.

Obviously, we could also define r-partitions for $r > 2$. Indeed, this has been done in [5]. But the accompanying notational complexity outweighs the advantage of the more general statements of the results to follow. Moreover, these developments using only 2-partitions are a sufficient basis for such generalizations [5].

III. Actual and Projected Coverages

In this paper we will use $T(\alpha_K)$ to denote the standard complete test set for the fault event $\alpha_K \in F_K$. This test set consists, of course, of all those input vectors to N which can detect α_K, and can be detailed, for example, as $T(\alpha_K) = \{X_i | Z[X_i; Y_K] \oplus Z[X_i; \alpha_K] = 1\}$. Given a set of input vectors, denoted as T, for N, we will be interested in the capability of T to cover (that is, detect) fault events in F_K. Accordingly, we will let $AC_K(T) \subseteq F_K$ denote the set of fault events in F_K which are actually covered by T. In other words, $\alpha_K \in AC_K(T)$ implies that $T \cap T(\alpha_K) \neq \emptyset$. We will in the following refer to the actual coverage of T for F_K as being complete when $AC_K(T) = F_K$. It should be kept in mind, however, that $AC_K(T)$ is usually unknown, and it is the goal of this note to consider means of determining or approximating $AC_K(T)$.

Accordingly, in addition to actual coverage, we will be very much concerned in the following with what we will refer to as the projected coverage of a fault complex by a test set. This projected coverage will be an estimate of the actual coverage. Hence, relative to the above, we will let $PC_K(T) \subseteq F_K$ denote the set of fault events in F_K which are estimated to be covered by T where $PC_K(T)$ will always be less than or equal to $AC_K(T)$. The process of calculating $PC_K(T)$ will be referred to as test set coverage projection and can generically be described as follows. In the network N, and for the set K of v lines in N, let I and J form a 2-partition of K. Let T now be a set of input vectors whose (either actual or projected) coverages for the component fault complexes F_I and F_J of F_K are known. Test set coverage projection can then be described as the process of projecting the coverage of T to F_K based upon our knowledge of the coverage of T for F_I and F_J. Accordingly, we will refer to the component fault complexes F_I and F_J as the basis of projection and to F_K as the target fault complex. In Section V, we will give a projection formula which can be used to project the coverage of a test set to a target fault complex.

It should be noted here that the resulting projected coverage by T of the target fault complex F_K would ideally be equal to the unknown actual coverage (that is, $PC_K(T) = AC_K(T)$). If, indeed, this is so, then in the following we will say that the projected coverage of T for F_K is exact. However, whether or not the projected coverage is exact clearly depends on a number of factors. Certainly one of the leading factors is whether our knowledge of the coverage of T for the basis of projection is, indeed, the actual coverage. But even if this is so, other factors such as the mechanics of the test set coverage projection and the relationships which exist among T and the fault events of F_I and F_J obviously also enter into this issue. This latter factor is the topic of the next section.

IV. Consistency and Testing Direction

In this section we will define two relationships between faults and test sets which will be seen to be central to test set coverage projection from a basis to a target.

Definition 4.1: A fault event $\alpha_j \in F_j$ is said to be a β-consistent fault event for an input vector X_i relative to the subfunction $H[X; Y_I]$ if $H[X_i; Y_I] = \beta$ implies $H[X_i; \alpha_j] = \beta$ where $\beta \in \{0, 1\}$.

Accordingly, given any $H[X; Y_I]$ and an input vector X_i to N, we can determine the set of fault events

$$F_I^\beta(X_i) = \{\alpha_j \mid H[X_i; Y_I] = H[X_i; \alpha_j] = \beta\}.$$

In other words, $F_I^\beta(X_i) \subseteq F_I$ is the set of all the β-consistent fault events of F_I for X_i relative to $H[X; Y_I]$. Along similar lines, we now give the following.

Definition 4.2: Given a set of input vectors T to N and the fault complex F_I, a fault event $\alpha_I \in F_I$ is said to be a $(0 \to 1)$-*tested fault* event for T if for some input vector $X_i \in T$, we have that $Z[X_i; Y_I] = 0$ and $Z[X_i; \alpha_I] = 1$.

In other words, if α_I is a $(0 \to 1)$-tested fault event for T, then there must exist at least one input vector $X_i \in T$ which makes the fault-free output of N a logical zero and the faulty output of N when the fault event α_I is present a logical one. In the following, such input vectors will be said to $(0 \to 1)$-test α_I. The above can similarly be stated for $(1 \to 0)$-testing.

Using Definition 4.2, we now supplement the notation introduced in Section III by letting $AC_I^{01}(T)$ and $AC_I^{10}(T)$, respectively, denote the set of fault events of F_I which are actually $(0 \to 1)$-tested and $(1 \to 0)$-tested fault events for T. Clearly, $AC_I^{01}(T) \cup AC_I^{10}(T) = AC_I(T)$. Similarly, we let $PC_I^{01}(T)$ and $PC_I^{10}(T)$, respectively, denote the set of fault events of F_I which are projected to be $(0 \to 1)$-tested and $(1 \to 0)$-tested fault events for T. Clearly, $PC_I^{01}(T) \cup PC_I^{10}(T) = PC_I(T)$.

Finally, given a fault event, say, $\alpha_I \in AC_I(T)$, or $\alpha_j \in PC_J(T)$, we will use $(T(\alpha_I) \cap T)^{10}$ to refer to the set of all those input vectors in T, each of which $(1 \to 0)$-tests α_I. $(T(\alpha_I) \cap T)^{01}$ can be interpreted analogously.

V. Projection Formula

In the last section, we defined two relationships, namely, consistency and testing direction, between fault events of component fault complexes and test sets. Based on these two relationships, we will now present a formula for obtaining the projected coverage of a test set for any set of lines in N. We will then prove a fundamental theorem which relates this projected coverage to the actual coverage.

Projection Formula: Given

1) a set K of lines in N and a 2-partition I and J of K; and

2) a test set T which is known to cover the set $PC_I(T)$ of fault events of F_I and the set $PC_J(T)$ of fault events of F_J;

then the projected set $PC_K(T)$ of the fault events of F_K covered by T is given as follows.

Part 1: If $Z[X; Y_K] = (H[X; Y_I] + M[X; Y_J]) + 0[X]$, then

$$PC_K(T) = PC_K^{01}(T) \cup PC_K^{10}(T)$$

where

$$PC_K^{01}(T) = (PC_I^{01}(T) \times F_J) \cup (F_I \times PC_J^{01}(T))$$

and

$$PC_K^{10}(T) = S_I^{10}(T) \cup S_J^{10}(T)$$

where

$$S_I^{10}(T) = \bigcup_{\alpha_j \in PC_J^{10}(T)} \left(\bigcup_{X_i \in (T \cap T(\alpha_j))^{10}} (\{\alpha_j\} \times F_I^0(X_i)) \right)$$

and

$$S_J^{10}(T) = \bigcup_{\alpha_j \in PC_J^{10}(T)} \left(\bigcup_{X_i \in (T \cap T(\alpha_j))^{10}} (F_I^0(X_i) \times \{\alpha_j\}) \right).$$

Part 2: If $Z[X; Y_K] = (H[X; Y_I] \cdot M[X; Y_J]) + 0[X]$, then the projection formula is the dual of Part 1.[1]

Although notationally complex in structure, the Projection Formula will be seen to be quite useful in the next section. Moreover, a rational for the structure of the Projection Formula (and its associated complexity) can be provided for the reader by taking the general perspective of path sensitization [7]. Note that the equation for $PC_K^{01}(T)$ suggests that: a) for every $\alpha_I \in PC_I^{01}(T)$, we will have that every $\alpha_K = (\alpha_I, \alpha_J)$, where $\alpha_J \in F_J$, will also be detected by T; and b) for every $\alpha_J \in PC_J^{01}(T)$, we will have that every $\alpha_K = (\alpha_I, \alpha_J)$, where $\alpha_I \in F_I$ will also be tested by T. With this in mind, if we refer now to the network of Fig. 2, which is one possible realization of the given $Z[X; Y_K]$, it is seen that because of the OR gate at the output, if a fault, say, α_I in subnetwork N_1 was $(0 \to 1)$-tested by means of sensitizing line 1, then no fault in N_2 could desensitize this line and mask fault α_I. Similarly, faults in subnetwork N_1 cannot mask $(0 \to 1)$-tested faults of N_2. The above, then, gives a rational for the structure of the equation for $PC_K^{01}(T)$ in the Projection Formula.

Similarly, consider the equation for, say, $S_I^{10}(T)$ relative to the network of Fig. 2. Assume that α_I is a fault in N_1 which is $(1 \to 0)$-tested by some X_i in T. In other words, for this X_i, line 1 has the logical value of 1 when α_I is not present and will be 0 otherwise, and line 2 will be 0 in both instances. Now, if a fault, say, α_J occurs in N_2 such that the value on line 2 remains at 0 for this X_i, then, clearly, X_i will also $(1 \to 0)$-test the fault $\alpha_K = (\alpha_I, \alpha_J)$. From Definition 4.1, such an F_J must belong to $F_J^0(X_i)$. Hence, all the faults in $\{\alpha_I\} \times \{F_J^0(X_i)\}$ will be detected by X_i. Moreover, the above would hold true for every $X_i \in T$ which $(1 \to 0)$-tests α_I. Hence, the following set of faults will be $(1 \to 0)$-tested by T:

$$\left(\bigcup_{X_i \in (T \cap T(\alpha_I))^{10}} (\{\alpha_I\} \times F_J^0(X_i)) \right).$$

Calculating such sets for all $\alpha_I \in PC_I^{10}(T)$ and unioning them provides the rational for the equation for $S_I^{10}(T)$.

It is important to note that the Projection Formula has an inherent, recursive nature in the sense that the projected coverage $PC_K(T)$ obtained by its use can then be used for determining the projected coverage of T to other fault complexes in N of which F_K is a component fault complex. It should also be pointed out that by virtue of this recursive nature, the Projection Formula can be stated more generally for r-partitions, $r > 2$ [5].

The following theorem is a crucial justification of the Projection Formula.

Theorem 5.1: For the Projection Formula,

$$PC_K(T) \subseteq AC_K(T).$$

Proof: We will prove this theorem only for the case in which $PC_K(T)$ is obtained by using Part 1 of the Projection Formula, that is, for the case where $Z[X; Y_K] = (H[X; Y_I] + M[X; Y_J]) + 0[X]$. Note that since the equations in Part 1 and Part 2 of the Projection Formula are duals, a proof of the theorem corresponding to Part 2 of the Projection Formula follows similarly. The following preliminaries will facilitate the proof.

Let α_K be a fault event of F_K. We know that $\alpha_K \in AC_K(T)$ if and only $T(\alpha_K) \cap T \neq \phi$. Hence, to show that for every $\alpha_K \in PC_K(T)$, $T(\alpha_K) \cap T \neq \phi$, we will study the composition and properties of $T(\alpha_K)$.

Now, there are many techniques by which $T(\alpha_K)$ can be

[1] (Note that since $0[x]$ does not explicitly play any role in the Projection Formula, the absence of exact duality in $Z[X; Y_K]$ in Parts 1 and 2 is not of importance in our considerations.)

detailed, but an algebraic or functional approach which will be most useful in our considerations is the Boolean difference method wherein

$$T(\alpha_K) = \{X_i | Z[X_i; Y_K] \oplus Z[X_i; \alpha_K] = 1\}.$$

Since in Part 1 of the Projection Formula we have that $Z[X; Y_K] = (H[X; Y_I] + M[X; Y_J]) + 0[X]$ and $\alpha_K = (\alpha_I, \alpha_J)$ for some $\alpha_I \in F_I$ and $\alpha_J \in F_J$, we can also express $T(\alpha_K)$ as follows:

$$T(\alpha_K) = \{X_i | ((H[X_i; Y_I] + M[X_i; Y_J]) + 0[X_i]) \oplus ((H[X_i; \alpha_I] + M[X_i; \alpha_J]) + 0[X_i]) = 1\}. \quad (1)$$

To show that $T(\alpha_K) \cap T \neq \phi$, $\alpha_K \in PC_K(T)$, we must determine $T(\alpha_I)$ and $T(\alpha_J)$, since out information about T is limited only to its coverage of the fault events F_I and F_J. As above, then, we can obtain $T(\alpha_I)$ and $T(\alpha_J)$ as follows:

$$T(\alpha_I) = \{X_i | Z[X_i; Y_I] \oplus Z[X_i; \alpha_I] = 1\} = \{X_i | ((H[X_i; Y_I] + M[X_i; Y_J]) + 0[X_i]) \oplus ((H[X_i \alpha_I] + M[X_i; Y_J] + 0[X_i]) = 1\} \quad (2)$$

and

$$T(\alpha_J) = \{X_i | Z[X_i; Y_J] + Z[X_i; \alpha_J] = 1\} = \{X_i | ((H[X_i; Y_I] + M[X_i; Y_J]) + 0[X_i]) \oplus ((H[X_i; Y_I] + M[X_i; \alpha]) + 0[X_i]) = 1\}. \quad (3)$$

With these details regarding the composition of $T(\alpha_K)$, $T(\alpha_I)$, and $T(\alpha_J)$ in mind, we can now proceed to the formalities of the proof. To show that $PC_K(T)$ obtained by using Part 1 of the Projection Formula is contained in or is equal to $AC_K(T)$, we will only consider the $PC_K^{10}(T)$ subset of $PC_K(T)$. The theorem can similarly be proven for the $PC_K^{01}(T)$ subset of $PC_K(T)$.

Let α_I be a fault event of F_I in $PC_I^{10}(T)$. Furthermore, let X_i be an input vector in T which $(1 \to 0)$-tests α_I, that is, $X_i \in (T \cap T(\alpha_I))^{10}$. By Definition 4.2 for this X_i we have that $Z[X_i; Y_I] = 1$ and $Z[X_i; \alpha_I] = 0$. But more particularly observe that from (2) this implies that $M[X_i; Y_J] = 0[X_i] = 0$, $H[X_i; Y_I] = 1$, and $H[X_i, \alpha_I] = 0$. Noting that we have $M[X_i; Y_J] = 0$, we now consider $F_J^0(X_i)$, the fault events of F_J which are 0-consistent relative to $M[X; Y_J]$ for X_i. Letting $\alpha_J \in F_J^0(X_i)$, we now direct our concern to the fault event $\alpha_K = (\alpha_I, \alpha_J)$ of F_K. Since, by the Definition 4.1, we have that $M[X_i; Y_J] = M[X_i; \alpha_J] = 0$, it should be clear that together with the above observations regarding (2), we have from (1) that $X_i \in T(\alpha_K)$, that is, $T(\alpha_K) \cap T \neq \phi$. Furthermore, since $Z[X; Y_K] = 1$ and $Z[X; \alpha_K] = 0$, we have also that $\alpha_K \in AC_K^{10}(T)$.

The above is evidently true for all fault events of F_K in $\{\alpha_I\} \times F_J^0(X_i)$ corresponding to every $X_i \in (T \cap T(\alpha_I))^{10}$. In other words,

$$\bigcup_{X_i \in (T \cap T(\alpha_I))^{10}} (\{\alpha_I\} \times F_J^0(X_i)) \subseteq AC_K^{10}(T). \quad (4)$$

Moreover, (4) is clearly true for every $\alpha_I \in PC_I^{10}(T)$. Thus, we have that

$$\bigcup_{\alpha_I \in PC_I^{10}(T)} \left(\bigcup_{X_i \in (T \cap T(\alpha_I))^{10}} (\{\alpha_I\} \times F_J^0(X_i)) \right) \subseteq AC_K^{10}(T). \quad (5)$$

For convenience, we refer to the left-side set of (5) as $S_I^{10}(T)$.

We can similarly show that

$$\bigcup_{\alpha_J \in PC_J^{10}(T)} \left(\bigcup_{X_i \in (T \cap T(\alpha_J))^{10}} (F_I^0(X_i) \times \{\alpha_J\}) \right) \subseteq AC_K^{10}(T). \quad (6)$$

As before, for convenience, we refer to the left-side set of (6) as $S_J^{10}(T)$.

Combining (5) and (6) yields

$$PC_K^{10}(T) = S_I^{10}(T) \cup S_J^{10}(T) \subseteq AC_K^{10}(T). \quad \text{Q.E.D.}$$

In other words, we have shown in Theorem 5.1 that the projected coverage $PC_K(T)$ obtained by using the Projection Formula is a lower bound on the actual coverage $AC_K(T)$. This means that the Projection Formula gives at worst an inexact coverage.

Consider now the particular case, wherein T is a complete single fault detection test set. In such a case, $PC_K(T)$ can be obtained by a recursive use of the Projection Formula. That is, if either $PC_I(T)$ or $PC_J(T)$ or both are not available, then the Projection Formula could first be used to calculate those unavailable projection sets. In fact, since we are dealing with a complete single fault detection test set, such a recursive use of the formula would continue until the component complexes being considered are associated with single lines of N. It might now seem that for large K, such recursive use of the Projection Formula could lead to a poor $PC_K(T)$ as an estimate to $AC_K(T)$, but such is not always the case, as is shown in the next section.

VI. Actual and Projected Coverages Revisited

In this section we will consider some cases to illustrate situations in which the projected coverage can be shown to be complete or exact relative to the actual coverage of the target fault complex.

We begin by giving the following theorem regarding completeness in the sense that situations can be specified wherein the projected coverage given by the Projection Formula will be the entire target fault complex, that is, $PC_K(T) = F_K$. Doing so will not only give further insight into the Projection Formula, but will also reinforce/extend the principal results available in the literature [1]–[4] regarding complete multiple fault coverage situations for single fault detection test sets.

Theorem 6.1:

a) Given any set K of lines and a complete single fault detection test set T_s for a cascaded two-level network, the projected coverage $PC_K(T_s)$ obtained from the Projection Formula equals F_K. (See [1].)

b) Given any set K of lines and a complete single fault detection test set T_s for an internal fanout-free network N, if the lines of K are inputs to three or fewer gates, the projected coverage $PC_K(T_s)$ obtained from the Projection Formula equals F_K. (See [2] and [3].)

c) Given any set K of lines and a complete single fault detection test set T_s for an internal fanout-free network N, if $Z[X; Y_K] = H[X; Y_I] * M[X; Y_J] + 0[X]$, $|I| = |K| - 1$, $|J| = 1$, and if T_s covers all the fault events of F_I, then the projected coverage $PC_K(T_s)$ obtained from the Projection Formula equals F_K. (See [4].)

Proof: Given in [5], and available from the authors.

We will now consider some of those situations wherein the Projection Formula is to be used, but the conditions of Theorem 6.1 are not satisfied. Our concern now will be exact projected coverage. We start with the following.

Definition 6.1: Given a set K of lines, a 2-partition I and J of K, and a set T of input vectors for N. If every $X_i \in T$ is such that it covers at least one fault event of $F_I \cup F_J$, or if every $X_i \in T$ which does not cover any fault event of $F_I \cup F_J$ is such that it does not cover any fault event of F_K as well, then T is said to satisfy the exactness requirement for F_I and F_J relative to F_K.

We will now prove a theorem which establishes a relation be-

tween $PC_K(T)$ and $AC_K(T)$ when T satisfies the exactness requirement.

Theorem 6.2: For the framework of the Projection Formula, if $PC_I(T) = AC_I(T)$, $PC_J(T) = AC_J(T)$, and T satisfies the exactness requirement for F_I and F_J relative to F_K, then $PC_K(T)$ obtained from the Projection Formula equals $AC_K(T)$.

Proof: We will prove this theorem (by contradiction) only for the case in which $PC_K(T)$ is obtained by using Part 1 of the Projection Formula as the other case follows similarly. We will also take advantage of the preliminaries established in the proof of Theorem 5.1.

Thus, assume that in F_K there exists a fault event

$$\alpha_K = (\alpha_I, \alpha_J) \in AC_K(T) = AC_K^{01}(T) \cup AC_K^{10}(T)$$

where

$$\alpha_K \in AC_K^{01}(T), \qquad \text{but } \alpha_K \notin PC_K^{01}(T) \tag{7}$$

or where

$$\alpha_K \in AC_K^{10}(T), \qquad \text{but } \alpha_K \notin PC_K^{10}(T). \tag{8}$$

We will now consider the implications of (8) and show that it leads to a contradiction. It can similarly be shown that (7) would also lead to a contradiction.

Suppose now that (8) is true. Then since $\alpha_K \in AC_K^{10}(T)$, consider any $X_i \in (T \cap T(\alpha_K))^{10}$. For this X_i we have that $Z[X_i; Y_k] = 1$ and $Z[X_i; \alpha_K] = 0$. But more particularly observe from (1) that this implies that

$$H[X_i; Y_I] + M[X_i; Y_J] = 1$$

and

$$H[X_i; \alpha_I] = M[X_i; \alpha_J] = 0[X_i] = 0.$$

Along these lines, suppose that $H[X_i; Y_I] = 1$ and $M[X_i; Y_J] = 0$. Then, clearly, $Z[X_i; Y_I] = 1$ and $Z[X_i; \alpha_I] = 0$, so $\alpha_I \in AC_I^{10}(T)$. Furthermore, since we have by hypothesis that $PC_I(T) = AC_I(T)$, then $\alpha_I \in PC_I^{10}(T)$. However, since $M[X_i; Y_J] = M[X_i; \alpha_J] = 0$, then $\alpha_J \in F_J^0(X_i)$. But this implies that $\alpha_K \in (\{\alpha_I\} \times F_J^0(X_i)) \subseteq PC_K^{10}(T)$, which contradicts (8). Similarly, if we assume that $H[X_i; Y_I] = 0$ and $M[X_i; Y_J] = 1$, we are also led to a contradiction of (8). Finally, the only remaining alternative is $H[X_i; Y_I] = M[X_i; Y_J] = 1$. But from (2) and (3) this implies that $X_i \notin T(\alpha_I)$ and $X_i \notin T(\alpha_J)$ for any $\alpha_I \in F_I$ or $\alpha_J \in F_J$, respectively. This, of course, contradicts the exactness requirement on T which states that every $X_i \in T$ covers at least one fault event in $F_I \cup F_J$, or if X_i does not cover any fault event of $F_I \cup F_J$, then X_i does not cover any fault of F_K. We therefore have that

$$AC_K^{10}(T) = PC_K^{10}(T). \qquad \text{Q.E.D.}$$

It is clear from the proof of Theorem 6.2 that if $PC_I(T) = AC_I(T)$ and $PC_J(T) = AC_J(T)$, then the projected coverage $PC_K(T)$ differs from the actual coverage $AC_K(T)$ only by those fault events of F_K which are covered by the input vectors in T, but which do not cover any fault event of F_I or F_J.

The result stated in Theorem 6.2 is useful for determining the actual coverage of a complete single fault detection test set for multiple faults by means of the Projection Formula. More particularly, given a set K of lines and a complete single fault detection test set T_s for N, suppose that we use the Projection Formula to determine the projected coverage $PC_K(T_s)$. Suppose further that for every execution of the Projection Formula algorithm, we have that the actual coverage of T_s for the projection basis is known, and the exactness requirement on T_s for the projection basis relative to the target complex is satisfied. Then, clearly, by Theorem 6.2 it can be concluded that $PC_K(T_s)$ so obtained would be the actual coverage $AC_K(T_s)$.

Example 2: Consider the network of Fig. 1(a) and the set K of 16 lines mentioned in Example 1. By using the Projection Formula with T as the set of input vectors given in Fig. 1(b), it can be shown that $|PC_K(T_s)| = 29\,822$. Furthermore, it can be seen that the conditions of Theorem 6.2 are, indeed, satisfied for every execution of the Projection Formula, and therefore we have that $PC_K(T) = AC_K(T)$.

When we consider those situations in which neither the conditions of Theorem 6.1 nor the exactness requirement is satisfied, we simply know that $PC_K(T)$ is a lower bound to $AC_K(T)$. First of all, it might be that this lower bound is quite close to complete coverage and, therefore, is acceptable in some applications. But more importantly, when we have the situation where the exactness requirement is not satisfied and when the projected coverage is too low to be acceptable with its uncertainty, then there is no recourse but to utilize some classical means of determining the actual coverage of T for F_K.

VII. Conclusion

In conclusion, our experience in using the Projection Formula on various internal fanout-free networks for various fault complexes [5], [6] has shown that the common assumption that the coverage capability of test sets is usually satisfactory for multiple faults on sizes greater than that for which test sets were explicitly generated must be used with discretion, as it has been so seen that even for such networks, the validity of this assumption is highly network structure/test set dependent. However, it seems to be possible to characterize the most difficult fault complexes for single fault detection test sets to cover by generic models [6]. This suggests that if such generic models could be avoided in the design structure of networks, the overall coverage capability would be enhanced. Indeed, in an extreme sense, this is the approach used in the results/networks reported in literature for which any complete single fault detection test sets cover all multiple faults [1].

References

[1] D. R. Schertz and G. Metze, "On the design of multiple fault diagnosable networks," *IEEE Trans. Comput.*, vol. C-20, pp. 1361-1364, Nov. 1971.

[2] R. J. Diephius, "Fault analysis for combinational logic networks," Ph.D. dissertation, Dep. Elec. Eng., Massachusetts Inst. Technol., Cambridge, Sept. 1969.

[3] J. W. Gault, J. P. Robinson, and S. M. Reddy, "Multiple fault detection in combinational networks," *IEEE Trans. Comput.*, vol. C-21, pp. 31-36, Jan. 1972.

[4] V. K. Agarwal and G. M. Masson, "A functional form approach to test set coverage in tree networks," *IEEE Trans. Comput.*, vol. C-28, pp. 50-52, Jan. 1979.

[5] ——, "Recursive coverage projection of test sets," Johns Hopkins Univ., Baltimore, MD, Elec. Eng. Rep. 77-11.

[6] ——, "Generic fault characterizations for table-look-up coverage bounding," *IEEE Trans. Comput.*, to be published.

[7] D. B. Armstrong, "On finding a nearly minimal set of fault detection tests for combinational logic sets," *IEEE Trans. Comput.*, vol. C-15, pp. 66-73, Feb. 1966.

[8] J. P. Hayes, "A NAND model for fault diagnosis in combinational logic networks, *IEEE Trans. Comput.*, vol. C-20, pp. 1496-1506, Dec. 1971.

[9] G. Markowsky, "A straightforward technique for producing minimal multiple fault test sets for fanout-free circuits," IBM Res. Rep. RC 6222, Yorktown Heights, NY, Sept. 29, 1976.

[10] G. Markowsky and C. W. Cha, "No single fault test set is smaller than any minimal multiple fault test set for a fanout-free combinational circuit," IBM Rep. RC 6483, Yorktown Heights, NY, Apr. 13, 1977.

[11] C. D. Latino and J. G. Bredeson, "Simplified multiple stuck-at-fault test generation techniques," in *Proc. 13th Annu. Allerton Conf.*, 1975, pp. 682-691.

[12] S. C. Seth and K. L. Kodandapani, "Diagnosis of faults in linear tree networks," *IEEE Trans. Comput.*, vol. C-26, pp. 29-33, Jan. 1977.

Multiple Fault Detection in Programmable Logic Arrays

VINOD K. AGARWAL

Abstract—The increasing recognition of PLA's as efficient and viable modules for such purposes as microprogramming and design of sequential controllers has led to a growing interest in the development of optimum fault detection test sets for these modules. It is now well known that a fault type which is unique to PLA's is the class of contact faults. A single contact fault is the spurious presence or absence of a contact between a row and a column of a PLA. We consider in this paper the problem of determining the capability of complete single contact fault test sets to cover multiple contact faults of PLA's. Our approach consists of developing a model of PLA's which allows one to represent a contact fault in a PLA as a stuck-at fault in the model of the PLA. Using this model, it is shown that more than 98 percent of all multiple contact faults of size 8 and less are inherently covered by every complete single contact fault test set in a PLA. Applications of this model to stuck-at fault diagnosis are also discussed.

Index Terms—Contact faults, masking, multiple fault detection, PLA fault detection, PLA modeling, programmable logic arrays, single fault coverage.

I. Introduction

A programmable logic array (PLA) is normally used to implement multioutput combinational logic by programming "blank" arrays of an AND-OR (or NOR-NOR) matrix [1]. The economy and flexibility accrued by using PLA's have moreover led to their growing usage in such areas as microprogramming, sequential controllers, function generators, and code conversion. The availability of field programmable logic arrays has furthermore provided the logic designer additional freedom to program an array on the site by blowing fusible links within the array. The reliability of these devices is therefore a matter of considerable importance [2]-[5].

Accordingly, this paper is concerned with the problem of fault detection (that is, fault testing) in PLA's. Since, conceptually, a PLA is simply a collection of many two-level AND-OR networks, it may be argued that a PLA can be tested by using the well-known techniques [6] for testing AND-OR networks. However the memory-like structure of a PLA not only leads to the usual stuck-at, bridge-type,

Manuscript received June 22, 1979; revised November 11, 1979. This work was supported in part by ONR Contract N00014-75-C-1196 and by the McGill University Faculty of Graduate Studies and Research Grant 943-81-19.

The author is with the Department of Electrical Engineering, McGill University, Montreal, P.Q., Canada.

0018-9340/80/0600-0518$00.75

and shorted-diode faults found in most combinational networks, but it also results in a unique type of contact faults not found in such networks. A contact fault (called a crosspoint fault in [2] and [3] and shrinkage and appearance faults in [4]) in a PLA is caused by the spurious presence or absence of a contact (that is, a fusible link) between a row and a column of the PLA. In a recent paper [2] by Ostapko and Hong, a scheme to generate a test set which detects the presence of every single contact fault in a PLA is presented. More importantly, it is also shown in their paper that any such test set for a PLA inherently covers (that is, detects) most of its single stuck-at, bridge-type, and shorted-diode faults as well. Similar results are likewise obtained in [3], [4].

Nonetheless, since the single fault model does not account for all the probable failures [7] of an LSI chip, it is clear that for high reliability applications, PLA's must be tested under a more general multiple fault model. But it is equally obvious that the consideration of various multiple faults which could possibly occur in a PLA would be extremely impractical. For example, in a PLA containing 16 input variables, 48 product terms, and 8 output lines, there are only 48 $(2 \times 16 + 8) = 1920$ single contact faults, but extremely large 2^{1920} multiple contact faults! Thus, as in the case of combinational networks, a practical compromise in such a situation would be to assume that a complete single contact fault test set for a PLA covers most of its multiple contact faults as well. The aim of our paper is to establish the validity of this assumption.

II. Contact Faults as Stuck-At Faults

The logic implemented by a PLA can be very conveniently described by a two-dimensional array of 0's and 1's, which is often referred to as the personality of the PLA [2]. The personality of a PLA which we will consider in the following is simply a representation of whether the contact between a row and a column of the PLA is a 0-contact or a 1-contact, where 0-contacts and 1-contacts are defined by the following.

Definition 1: Given a row and an *output* column in a fault-free PLA, a 1-contact (0-contact) will be said to exist between the row and the column if the link between the two is intact (correspondingly, is not intact, that is, electrically fused).

Definition 2: Given a row and an *input* column in a fault-free PLA, a 0-contact (1-contact) will be said to exist between the row and the column if the link between the two is intact (correspondingly, is not intact).

A two-input PLA with two rows and two output functions is shown in Fig. 1(a). Using the above definitions, the personality shown in Fig. 1(b) is obtained. Finally, Fig. 1(c) illustrates the equivalent logic path of this PLA.

We now formally define a single contact fault.

Definition 3: A single 0-contact (1-contact) fault is said to exist in a PLA if due to some failure, a 0-contact (1-contact) of the fault-free PLA becomes a 1-contact (0-contact) in the faulty PLA.

Accordingly, the total number of single contact faults which could possibly occur in a PLA would simply be the number of entries in its personality. For a single-input decoder PLA with n inputs, m rows, and p output functions, this number is easily seen to be $m(2n + p)$. Moreover, by generalizing Definition 3 to multiple contact faults, the total number of multiple contact faults of a given size, say $r > 1$, is seen to be $\binom{m(2n+p)}{r}$, and the total number of all single and multiple contact faults is

$$\sum_{r=1}^{m(2n+p)} \binom{m(2n+p)}{r} = 2^{m(2n+p)} - 1.$$

Assume that T_c denotes a complete single contact fault detection test set for a PLA; that is, T_c is such that for each single contact fault in the PLA, there exists at least one input vector in T_c for which the fault-free output of the PLA, on at least one output line, is different from the output in the presence of the fault. Similarly, let T_m denote a complete single and multiple contact fault detection test set for a PLA. As pointed out in the previous section, the generation of an optimal T_m is a rather impractical task. Therefore, our concern in this paper is to determine the capability of a T_c for a PLA to cover its multiple contact faults as well. We do this by developing a combinational network model of the PLA such that each single contact fault in it is functionally equivalent to some single stuck-type fault in the modeled network. This model will be referred to as the SAE network (stuck-at equivalent network) of the PLA.

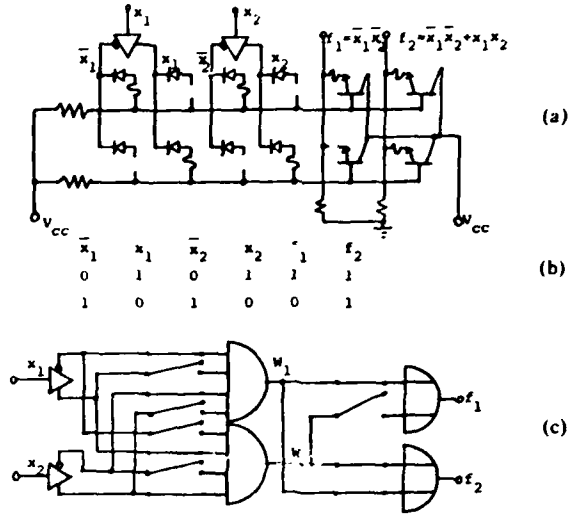


Fig. 1. An AND-OR PLA example.

The SAE network for a PLA is best obtained by considering contact faults on input and output columns in separate manners. Accordingly, consider first the simple AND network shown in Fig. 2(a), and assume that input lines x and y correspond to some input columns of a PLA and the gate output line W corresponds to some row of the PLA. Since each contact on lines x and y can be in two different positions, there are four different "programming" configurations associated with this AND gate, as shown in Fig. 2(a). To be able to represent a contact fault as a stuck-at fault, we will assume that each contact of a PLA is represented by an input Boolean variable in its SAE network, and that its value, 0 or 1 (as determined by Definition 2), in the PLA is the value of the line which carries that Boolean variable in the SAE network. More particularly, suppose a_x is a Boolean variable whose value is 0(1) if line x has a 0-contact (1-contact). Let b_y be similarly defined with respect to line y. With the introduction of these two new variables, W then becomes a function of x, y, a_x, and b_y, as shown in the K-map of Fig. 2(b). It is important to note here that a_x and b_y do not depend on the values of x and y, but only on the contacts on those lines. Therefore, x, y, a_x, and b_y are four independent variables. Using now the straightforward Karnaugh map technique, we get

$$W = (a_x + x)(b_y + y). \tag{1}$$

A network realizing (1) is shown in Fig. 2(c).

Comparing the network of Fig. 2(c) with those of Fig. 2(a), we note that depending on the values of a_x and b_y, the former network can be made equivalent to any one of the four "programming" configurations of the later networks. For example, for the top-left configuration of Fig. 2(a), the equivalent network is obtained by assigning $a_x = 1$ and $b_y = 1$. More importantly, this comparison also shows that a 0-contact (1-contact) fault on, say line x of a "programming" configuration, can be represented in its equivalent network by simply assigning a stuck-at-1 (stuck-at-0) fault to line a_x. In other words, we have shown that the SAE network for a PLA's input columns can be easily obtained by replacing each AND gate of the PLA with a corresponding network of the type shown in Fig. 2(c). This replacement in the example PLA of Fig. 1 results in the left part of Fig. 4.

Similar to the above, the stuck-at fault representation of contact faults on the output columns of a PLA is obtained by considering the OR gate network shown in Fig. 3(a). Assume here that input lines W_1 and W_2 correspond to some rows of a PLA and the gate output f corresponds to some output line of the PLA. Moreover, as above, let

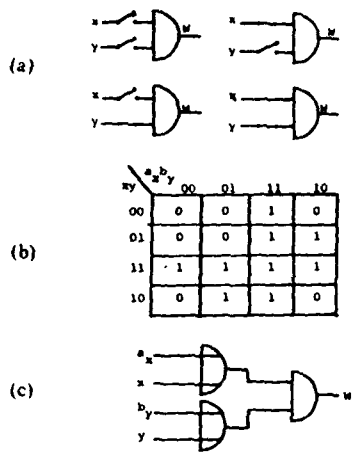


Fig. 2. SAE network for AND gate.

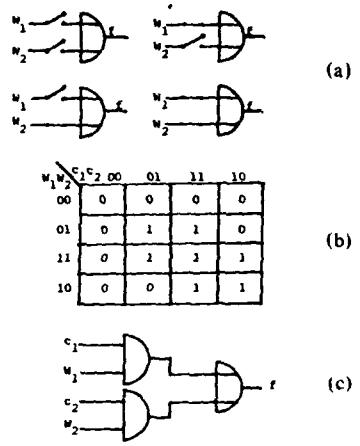


Fig. 3. SAE network for OR gate.

c_1 and c_2 denote Boolean variables whose value is 0(1) if lines W_1 and W_2, respectively, have a 0-contact (1-contact). Then the K-map [see Fig. 3(b)] for f in terms of W_1, W_2, c_1, and c_2 leads to the function

$$f = c_1 W_1 + c_2 W_2 \tag{2}$$

whose realization is shown in Fig. 3(c).

Here again, one easily sees that depending on the values of c_1 and c_2, each of the four "programming" configurations of Fig. 3(a) has an equivalent network in Fig. 3(c). Also, as above, a 0-contact (1-contact) fault on, say, line W_1 of a "programming" configuration is represented in its equivalent network as a stuck-at-1 (stuck-at-0) fault on the c_1 line. Based on these developments, the SAE network for the example PLA of Fig. 1 finally results in the network of Fig. 4.

Some important points should be mentioned here before we use this SAE network model for multiple fault coverage analysis in the next section. First of all, note from Fig. 4 that although the SAE network contains fanout both at primary input lines and internal lines, the internal line fanout does not reconverge. Therefore, when seen with respect to any one output line, the corresponding subnetwork of an SAE network is clearly an internal fanout-free network [12], [13] consisting only of three levels (because now levels 2 and 3 can be merged into one level).

Second, note that an SAE network contains two types of primary inputs: the *n regular* inputs $x_1, x_2, \cdots, x_n$, and $m(2n + p)$ *contacts inputs* which correspond to $m(2n + p)$ contacts of the associated PLA. In the fault-free SAE network, each of these $m(2n + p)$ contact inputs is assigned a permanent 0 or 1 depending on the value of the

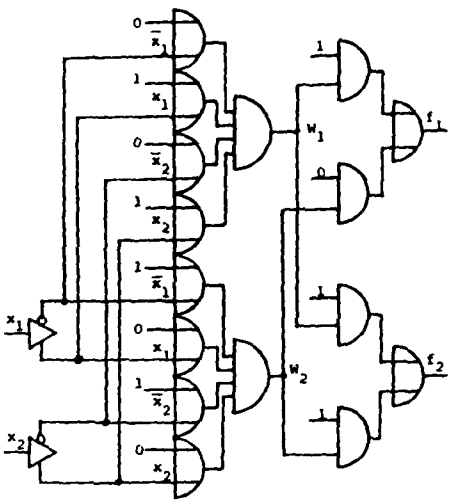


Fig. 4. The SAE network for the PLA of Fig. 1.

corresponding contact in the PLA. Thus, it should be clear that each contact input line with the value 0(1) could contain only one fault of interest, namely, sa1 (sa0), and that the other fault, sa0 (sa1), on that line would be meaningless. The resulting $m(2n + p)$ single stuck-at faults of interest on these $m(2n + p)$ contact input lines would, of course, correspond to the $m(2n + p)$ single contact faults of the PLA. Moreover, a multiple contact fault on a PLA would thus be represented by a multiple stuck-at fault on the contact input lines. For the convenience of description in the remainder of this paper, we will, whenever there is no change of ambiguity, use the terms "contact fault(s)" and "stuck-at fault(s) on contact input lines" interchangably.

Finally, it is important to note that although an SAE network of a PLA is far more complex than its equivalent logic network [compare Figs. 4 and 1(c)], a significant difference between the two representations is that whereas the former network represents the complete PLA structure, the latter network represents only the logic realized by the PLA.

III. Multiple Contact Fault Coverage

As pointed out in the last section, each SAE network output line is the output of an internal fanout-free combinational network. This observation immediately suggests the possibility of using various previously known results [8]–[11] about multiple stuck-at fault coverage in internal fanout-free networks to determine the multiple contact fault coverage in PLA's. For example, it is well known that in an irredundant, internal fanout-free network, every multiple stuck-at fault of size 2 and 3 is covered by any test set that covers all the single stuck-at faults of the network. Therefore, we immediately get the following from this well-known result.

Lemma 1: Each T_c of an irredundant PLA covers every multiple contact fault of size 2 and 3.

For multiple stuck-at faults of sizes greater than 3, it is shown in [11] that the greatest lower bound on the capability of single stuck-at test sets to cover multiple faults can be obtained by a simple table look-up process. Thus, similar to Lemma 1, this result could be used to determine the greatest lower bound on the multiple contact fault coverage capability of a T_c in a PLA. However, we now show in the following that it is possible to obtain better multiple contact fault coverage bounds by using the concept of masking than by applying previously known results.

Definition 4: Given a row, say W_i, and an input variable, say x_j, let the contact between row W_i and column $\bar{x}_j$ be denoted by a_{ij} and let the contact between row W_i and column x_j be denoted by b_{ij} where a_{ij} and $b_{ij} \in \{0, 1\}$. Then $A_{ij} = (a_{ij} + \bar{x}_j)(b_{ij} + x_j)$ will be referred to as an *internal variable* for W_i in terms of x_j.

An internal variable A_{ij} is simply a function of two contact input

variables a_{ij} and b_{ij} and one regular input variable x_j. Using this definition, the product function realized by each row can be conveniently written as

$$W_i = A_{i1}A_{i2} \cdots A_{in} \tag{3}$$

where $1 \leq i \leq m$.

Similar to Definition 4, let the contact between an output column f_k, $\leq k \leq p$ and a row W_i be denoted by c_{ki} where $c_{ki} \in \{0, 1\}$. Then the output f_k can be written as

$$\begin{aligned} f_k &= c_{k1}W_1 + c_{k2}W_2 + \cdots + c_{km}W_m \\ &= c_{k1}(A_{11}A_{12} \cdots A_{1n}) + c_{k2}(A_{21}A_{22} \cdots A_{2n}) \\ &\quad + \cdots + c_{km}(A_{m1}A_{m2} \cdots A_{mn}). \end{aligned} \tag{4}$$

Definition 5: A single stuck-at fault, say α_1, is said to be masked by another single stuck-at fault, say α_2, for an input vector X_t if X_t tests α_1, but does not test the simultaneous multiple fault $\alpha_1\alpha_2$.

With the help of above definitions, Lemmas 2 and 3 specify all the necessary conditions under which masking takes place in a PLA.

Lemma 2: A 0-contact fault in a row of a PLA can be masked only by a 1-contact fault in the same row.

Proof: Let α_1 be a 0-contact fault, say on row 1 and column $\bar{x}_1$ That is, $A_{11} = (a_{11} + \bar{x}_1)(b_{11} + x_1)$ where $a_{11} = 0$ and the fault α_1 causes A_{11} to become $(1 + \bar{x}_1)(b_{11} + x_1)$. Since the $a_{ij} = b_{ij} = 0$ combination is never allowed, we must have that $b_{11} = 1$. Assume further now that this fault α_1 is tested by an input vector $X_t \in T_c$ through the output line, say f_1. Then since

$$f_1 = c_{11}W_1 + c_{12}W_2 + c_{13}W_3 + c_{1m}W_m$$
$$\text{and } W_1 = A_{11}A_{12}, \cdots, A_{1n}.$$

X_t can test α_1 iff for input X_t, we have

$$c_{12}W_2 = c_{13}W_3 = \cdots = c_{1m}W_m = 0$$
$$c_{11} = 1$$
$$A_{12} = A_{13} = \cdots = A_{1n} = 1$$

and $A_{11} = (0 + \bar{x}_1)(1 + x_1) = 0$ for the fault-free case and $A_{11} = (1 + \bar{x}_1)(1 + x_1) = 1$ in the presence of α_1.

Let α_2 be any other single contact fault in PLA. We now show that unless α_2 is a 1-contact fault on row 1, it cannot mask α_1. To begin, suppose that α_2 is any contact fault on row i, $i \neq 1$. Then since $c_{1i}W_i = 0$ in the fault-free case, the presence of α_2 would at the worst make $c_{1i}W_i = 1$. But this would imply that $f_1 = 0$ in the fault-free case and $f_1 = 1$ in the presence of the fault $\alpha_1\alpha_2$, i.e., X_t would also test the double fault $\alpha_1\alpha_2$. Thus, α_2 would not mask α_1 if α_2 is on any row i, $i \neq 1$.

Suppose next that α_2 is a 0-contact fault on row 1. If this α_2 were on row 1 and any one of the output columns 2, 3, $\cdots$, p, then it is clear that the presence of α_2 woule never mask the testing of α_1 through f_1. Similarly, if α_2 were on row 1 and any input column, say j, then we would have that the faulty $A_{1j} = 1$ and, hence, that α_2 does not mask α_1.

Thus, we have shown that unless α_2 is a 1-contact fault on row 1, it cannot mask α_1. Even though this proof is given only for the case when α_1 is on an input column, a similar proof for α_1 on an output column can be easily provided. Q.E.D.

Lemma 3: A 1-contact fault in a row of a PLA can be masked only by a 0-contact fault in some other row.

Proof: The lemma can be proven in a similar manner to Lemma 2.

These two lemmas are important because they specify all the conditions under which one contact fault may be masked by another contact fault in a PLA. Using these lemmas, the proof of Lemma 1 is straightforward [15].

Although Lemma 1 ensures that each T_c covers all faults of sizes 2 and 3, a multiple contact fault of size 4 or larger might not be covered by each T_c. In fact, Fig. 5 contains an example of a simple PLA where four contact faults α_1, α_2, α_3, α_4 are such that α_1 masks α_4, α_4 masks α_3, α_3 masks α_2, and α_2 masks α_1. Therefore, the multiple

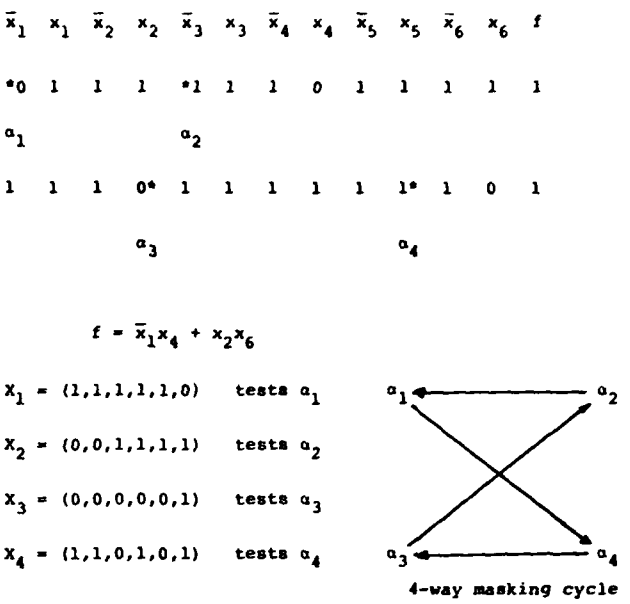


Fig. 5. PLA with four-way masking cycle.

fault $\alpha_1\alpha_2\alpha_3\alpha_4$ is not detected by the four input vectors, shown in Fig. 5, which detect the single contact faults α_1, α_2, α_3, and α_4. A similar observation about multiple stuck-at faults of size 4 is well known in the literature [9], [11].

This phenomenon of four-way masking cycle, which requires two pairs of a 0-contact fault and a 1-contact fault on two separate rows, is very crucial to the undetectability of a multiple contact fault by a T_c in an irredundant PLA.

Theorem 1: Every multiple contact fault in an irredundant PLA is detected by any T_c provided the multiple fault does not contain pairs of a 0-contact fault and a 1-contact fault on more than one row.

Proof: Clearly, if a multiple fault indeed contains pairs of a 0-contact fault and a 1-contact fault on only one row, say i, then every other row in the PLA which contains component single contact faults of the multiple fault must contain either all the 0-contact faults or all the 1-contact faults.

Now since row i contains a 0-contact and a 1-contact fault pair, it might be possible, by Lemma 2, that the 0-contact fault is masked by the 1-contact fault. Moreover, by Lemma 3, this 1-contact fault on row i might, in turn, be masked by some other 0-contact fault in some other row, say $j \neq i$. However, since by our assumption row j does not contain any 1-contact fault, it is clear from Lemma 2 that this 0-contact fault cannot be masked by any single contact fault component of the given multiple fault. In fact, it can furthermore be seen from the proof of Lemma 2 that this 0-contact fault cannot even be masked by any multiple contact fault component of the given multiple fault. Therefore, any test in T_c which tests this 0-contact will test the given multiple contact fault as well. Q.E.D.

Although the condition in Theorem 1 is only a sufficient condition, it is easy to show that most multiple faults of interest, such as faults of size 8 and less [7], are included in this condition. We begin by showing that Theorem 1 implies that most multiple contact faults of size 4 are covered by each T_c.

Theorem 2: Out of the total $\binom{m(2n+p)}{4}$ different contact faults of size 4 in an irredundant PLA, at the most $\binom{m}{2} \cdot (n + p/2)^4$ faults are not covered by every T_c of the PLA.

Proof: By Theorem 1, every multiple fault of size 4 which does not form the four-way masking cycle is bound to be covered by each T_c. So Theorem 2 would immediately follow if we show that the maximum number of contact faults of size 4 with four-way masking is $\binom{m}{2} \cdot (n + p/2)^4$.

Let $\alpha_1\alpha_2\alpha_3\alpha_4$ be one such contact fault. Since it must exist on exactly some 2 of the m rows of the PLA, there are $\binom{m}{2}$ different pairs

of two rows which are possible locations of $\alpha_1\alpha_2\alpha_3\alpha_4$. Let W_{i1} and W_{i2} be one of these $\binom{m}{2}$ pairs, and assume that α_1 and α_2 exist on W_{i1} and α_3 and α_4 exist on W_{i2}. The total number of contacts on each row is, of course, $2n + p$. Let the number of 0-contacts and 1-contacts, respectively, on row W_{i1} be s_1 and $2n + p - s_1$, and on row W_{i2} be s_2 and $2n + p - s_2$ where $s_1 > 0$ and $s_2 > 0$. Assuming now that α_1 and α_3 are 0-contact faults and α_2 and α_4 are 1-contact faults, we note that all the different ways of having such α_1, α_2, α_3, and α_4 on W_{i1} and W_{i2} are

$$\binom{s_1}{1}\cdot\binom{2n+p-s_1}{1}\cdot\binom{s_2}{1}\cdot\binom{2n+p-s_2}{1} = s_1(2n+p-s_1)(s_2)(2n+p-s_2). \quad (5)$$

Clearly, the maximum value of the expression in (5) results when

$$s_1 = s_2 = \frac{2n+p}{2} = n + \frac{p}{2}.$$

It thus follows that the maximum number of contact faults of size 4 with four-way masking is $\binom{m}{2}\cdot(n + p/2)^4$. Q.E.D.

For $m = 48$, $n = 16$, and $p = 8$, the number $\binom{m}{2}\cdot(n + p/2)^4$ is a mere 0.03 percent of the total $\binom{m(2n+p)}{4}$. Thus, 99.97 percent of all multiple contact faults of size 4 are bound to be covered by each T_c in a typical irredundant PLA.

Theorem 2 can now be generalized for multiple faults of larger sizes, as follows. For the proof of this theorem, see [15].

Theorem 3: Out of the total $\binom{m(2n+p)}{r}$, $r \geq 4$, different faults of size r in an irredundant PLA, at the most $\binom{m}{2}\cdot(n + p/2)^4 \cdot \binom{m(2n+p)-4}{r-4}$ faults are not covered by every T_c of the PLA.

Note that the bound given in Theorem 3 is simply an upper bound. In fact, for large values of r, the bound $\binom{m}{2}\cdot(n + p/2)^4 \cdot \binom{m(2n+p)-4}{r-4}$ becomes greater than $\binom{m(2n+p)}{r}$. However, for small values of r, Theorem 3 is very convenient to use. For example, when $m = 48$, $n = 16$, and $p = 8$, Theorem 3 shows that more than 98 percent of all multiple contact faults of size 8 and less in an irredundant PLA are covered by each T_c of the PLA. This result becomes very significant in the light of a recent paper [7], wherein it is shown that the consideration of faults of size 8 and less is sufficient for most LSI chips. Thus, in conclusion, we have shown that even though contact faults in PLA's exhibit the well-known masking phenomenon, yet more than 98 percent of all multiple contact faults of interest are covered by their single contact fault test sets.

IV. Conclusion

The problem of multiple fault coverage by single contact fault test sets in PLA's has been considered in this paper. Since a PLA is conceptually a two-level network, it may seem that each complete single contact fault test set for a PLA must inherently cover all of its multiple contact faults as well. However, in Section II an SAE (stuck-at equivalent) network of a PLA was developed to show that such is not the case in general. Nonetheless, it is also shown by using SAE networks that more than 98 percent of all multiple contact faults of interest are, indeed, covered by every single fault test set in an irredundant PLA.

The results mentioned above have been developed for AND-OR PLA's with single input decoders only. However, similar results for NOR NOR PLA's with two input decoders are easily derived. More specifically, the personality for a NOR NOR PLA is determined by using Definition 1 for both input and output columns. Moreover, since a NOR gate is an OR followed by a NOT, the SAE network for such a PLA is obtained by using the network of Fig. 3. Finally, an internal variable A_{ij} is then defined as $A_{ij} = a_{ij}\bar{x}_j\bar{x}_{j+1} + b_{ij}\bar{x}_j x_{j+1} + c_{ij}x_j\bar{x}_{j+1} + d_{ij}x_j x_{j+1}$ where $i = 1, 2, \cdots, m$ and $j = 1, 3, \cdots, n-1$. Using these formulations, results similar to Lemmas 1-4 and Theorems 1-3 of Sections II and III can be easily proven.

In the above analysis, we had tacitly assumed that each single contact fault of a PLA is detectable. But, in practice, most PLA's tend to contain various undetectable faults. The multiple fault coverage capability of such PLA's, by their single fault test sets, is generally reduced. This claim is best illustrated by considering a double contact fault, say a 0-contact fault and a 1-contact fault, on some row of a PLA. By Lemmas 2 and 3, we then know that even if the 1-contact fault masked the 0-contact fault, the converse is never true. Thus, the double fault is sure to be covered by a complete single fault test set as long as the 1-contact fault is detectable. However, if the 1-contact fault is undetectable and it masks the 0-contact fault, then the double fault might not be detected by a complete single fault test set.

The above illustration can similarly be carried out for multiple faults of larger sizes. Thus, it seems evident that the multiple fault coverage capability of a T_c in a redundant PLA will be highly dependent on the personality of the PLA and the redundant contacts. In other words, no general results like Theorems 1 to 3 of the last section can be specified for redundant PLA's. Nonetheless, one easily notices from the proof of Theorem 1 that any multiple contact fault, which contains at least one detectable 0-contact fault component in any row containing only 0-contact fault components, would always be covered by each T_c. Further work along these lines, however, needs to be done.

Finally, the advantage of the SAE network models of PLA's in areas other than the multiple contact fault coverage problem is worth mentioning here. Note, for instance, that a T_c is basically an incomplete, but very specific single stuck-at fault test set for a SAE network. Thus, by taking into account the explicit structure possessed by all SAE networks, it can be easily proven that a T_c in general will cover most of the single stuck-at faults of a SAE network. Moreover, it will then immediately follow that each T_c also covers most stuck-at faults of sizes up to 8 [11]. In short, a T_c for a PLA can be considered a very effective test set for all single and multiple contact and stuck-at faults of interest.

Acknowledgment

The author would like to express his sincere gratitude to Prof. G. M. Masson for his support and encouragement throughout this work.

References

[1] H. Fleisher and L. I. Maissel, "An introduction to array logic," *IBM J. Res. Develop.*, vol. 19, pp. 98-109, Mar. 1975.
[2] D. L. Ostapko and S. J. Hong, "Fault analysis and test generation for programmable logic arrays (PLA)," in *Proc. FTCS-8*, France, 1978.
[3] C. W. Cha, "A testing strategy for PLAs," IBM Thomas J. Watson Res. Cen., Yorktown Heights, NY 10598, Res. Rep. RC 6832 (#29313), Nov. 1977.
[4] J. E. Smith, "Detection of faults in programmable logic," IEEE Comput. Soc. Repository, R-78-2, IEEE Comput. Soc., Long Beach, CA.
[5] E. I. Muelhldorf and T. W. Williams, "Optimized stuck fault test pattern generation for PLA macros," in *Dig. Semiconductor Test Symp.*, Cherry Hill, NJ, IEEE Catalog Number 77CH126-7C, Oct. 25-27, 1977, pp. 88-101.
[6] M. A. Breuer and A. D. Friedman, *Diagnosis and Reliable Design of Digital Systems*. Woodland Hills, CA: Computer Science Press, 1976.
[7] L. H. Goldstein, "A probabilistic analysis of multiple faults in LSI circuits," IEEE Comput. Soc. Repository, R-77-304, IEEE Comput. Soc., Long Beach, CA.
[8] D. R. Schertz and G. Metze, "On the design of multiple fault diagnosable networks," *IEEE Trans. Comput.*, vol. C-21, pp. 13-36, Jan. 1972.
[9] J. W. Gault, J. P. Robinson, and S. M. Reddy, "Multiple fault detection in combinational networks," *IEEE Trans. Comput.*, vol. C-21, pp. 31-36, Jan. 1972.
[10] C. T. Ku and G. M. Masson, "The Boolean difference and multiple fault analysis," *IEEE Trans. Comput.*, vol. C-24, pp. 62-71, Jan. 1975.
[11] V. K. Agarwal and G. M. Masson, "Generic fault characterizations for table-look-up coverage bounding," *IEEE Trans. Comput.*, vol. C-29, Mar. 1980.
[12] J. P. Hayes, "The fanout structure of switching functions," *J. Ass. Comput. Mach.*, vol. 22, pp. 551-571, Oct. 1975.
[13] V. K. Agarwal, "Fanout-free Boolean functions and L-expressions," in *Proc. 1978 Conf. Inform. Sci. Syst.*, The Johns Hopkins Univ., Baltimore, MD, 1978, pp. 227-233.
[14] D. C. Bossen and S. J. Hong, "Cause effect analysis for multiple faults in combinational networks," *IEEE Trans. Comput.*, vol. C-20, pp. 1252-1257.
[15] V. K. Agarwal, "Multiple fault detection in programmable logic arrays," Dep. Elec. Eng., McGill Univ., Montreal, P.Q., Canada, Rep. 79-4

Generic Fault Characterizations for Table Look-Up Coverage Bounding

VINOD K. AGARWAL, SENIOR MEMBER, IEEE, AND GERALD M. MASSON, SENIOR MEMBER, IEEE

Abstract—**Given any combinational, internal fan-out-free network and any complete single fault detection test set (SFDTS) for the network, we consider in this paper the problem of determining the minimal extent to which that SFDTS will cover multiple faults in the network. The basis of our approach is the development of a generic perspective to multiple faults which uses a representation of such faults called an *L*-expression. This perspective leads to a technique for obtaining the greatest lower bound on the multiple fault coverage capability of an SFDTS by means of a simple table look-up process. In addition to generalizing previously known results regarding multiple fault coverage, two particularly interesting results obtained from this approach are as follows:**

1) On the average, every SFDTS for an internal fan-out-free network covers 92 percent of all multiple faults of sizes 8 and less.

2) On the average, every SFDTS for an internal fan-out-free network covers at least 46.1 percent of all multiple faults.

Index Terms—**Coverage bounds, coverage table, fault vectors, generic representations, *L*-expressions, internal fan-out-free networks, single and multiple fault detection.**

I. Introduction

THE increasing density of logic on integrated circuit chips together with the emerging LSI design practice [23] of implementing sequential networks in the form of register to register combinational logic have led to a renewed interest in various unsolved problems associated with the detection of stuck-type faults in combinational networks. Perhaps the most basic of such problems is the generation of tests which efficiently detect the presence of multiple faults in a given combinational network. The complexity of this problem stems from the fact that in a network of p lines, there are 3^p-1 different multiple faults which might exist. To combat this prohibitively large number of faults, various concepts such as functional equivalence [24] and fault collasping [25] have been developed. Testing algorithms [26], [27] based on these concepts show that instead of considering 3^p-1 multiple faults, it is possible to consider a smaller number of faults while still being able to detect the presence of any of the 3^p-1 faults. However, even the use of these algorithms does not provide a satisfactory solution for large networks.

A practical compromise often employed to alleviate this multiple fault detection problem is to use a complete single fault detection test set (SFDTS) for a network to cover its multiple faults as well. However, again because of the large number of multiple faults involved, this compromise has had little theoretical justification or investigation other than the establishment of such results as an SFDTS is a multiple fault detection test set (MFDTS) for two-level networks or every SFDTS in an internal fan-out-free network covers all multiple faults of sizes 2 and 3 [10]-[13]. Motivated by the above, the authors previously introduced the concepts of the functional form of multiple faults [9] and of recursive coverage projection of test sets [10] to theoretically investigate more general situations. These developments led to interesting extensions of previously known results [11]-[13] and new characterizations of some special cases of multiple faults. In this paper we continue this work by developing a general approach to quantitatively evaluate the validity of the common, practical compromise mentioned above. This approach leads to a technique for obtaining the greatest lower bound on the multiple fault coverage capability of a SFDTS on a set of lines by means of a simple table look-up process. Two particularly interesting results obtained using this table are as follows.

1) On the average, every SFDTS for an internal fan-out-free network covers 92 percent of all multiple faults of sizes 8 and less.

2) On the average, every SFDTS for an internal fan-out-free network covers at least 46.1 percent of all multiple faults.

II. Network Model and Notational Preliminaries

A. Network Model

For multiple fault detection purposes, it is convenient to classify a single output combinational switching network into one of the following three classes:

C1: Class of Fan-Out-Free Networks [1]-[3], [28]: A network in which no line fans out to two or more lines is referred to as a fan-out-free network [see Fig. 1(a)].

C2: Class of Internal Fan-Out-Free Networks [8]-[11]: In an internal fan-out-free network, fan-out is allowed on the primary input lines only [see Fig. 1(b)].

C3: Class of Reconvergent-Fan-Out Networks [12]-[15]: A network from this class can have fan-out on any line [see Fig. 1(c)].

Clearly, $C1 \subset C2 \subset C3$. The reason this classification scheme is centered around the amount of fan-out present in a network is that fan-out is the single most important factor which affects the complexity of generating test sets for networks. Indeed, networks of Class *C1* are the simplest to test and networks of Class *C3* are the most difficult. In fact, very simple and useful methods to generate MFDTS's for networks of Class *C1* are well known [4]-[7] in the literature. On the

Manuscript received August 4, 1977; revised August 13, 1979. This work was supported by the Office of Naval Research under Contract N00014-75-C-1196.

V. K. Agarwal is with the Department of Electrical Engineering, McGill University, Montreal, P.Q., Canada.

G. M. Masson is with the Department of Electrical Engineering, The Johns Hopkins University, Baltimore, MD 21218.

0018-9340/80/0400-0288$00.75

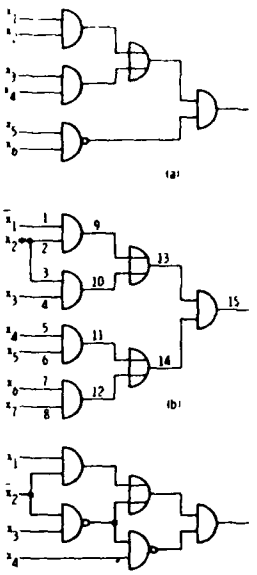


Fig. 1. Illustration of three classes of networks.

other hand, it is equally well known that the generation of an MFDTS for a complex network of Class *C3* can be a prohibitively difficult task.

In terms of the capability of each of these classes of networks to realize Boolean functions, it is known [1], [3] that functions which can be realized by Class *C1* are very limited. However, it can be shown that Classes *C2* and *C3* are complete in the sense that every Boolean function can be realized by a network of Class *C2* and, therefore, also of Class *C3*. The advantage of Class *C3* networks over Class *C2* networks is, of course, that the former type of networks require fewer gates for the realization of same functions. However, since Class *C2* is a complete class in the sense just mentioned above and since Class *C2* networks are easier to manipulate than Class *C3* networks from a multiple fault detection point of view, we will restrict our considerations in this paper to networks of Class *C2* only. This is not a severe limitation because each network of Class *C3* can be converted to a network of Class *C2* [13]. Hence, our results can be exploited for the networks of Class *C3* as well.

Thus, let N denote an internal fan-out-free network such as that shown in Fig. 1(b). For the convenience of analysis in this paper, we will make two simplifying assumptions on the line faults which can exist in such networks. First of all, since a stem fault [such as the one shown in Fig. 1(b)] is functionally equivalent to some branch faults (lines 2 and 3 s-a-1), we will assume that each multiple fault under consideration is such that it does not include any stem faults. Second, since two or more simultaneous single faults (that is, one multiple fault) on the same path from a primary input to the primary output are collectively functionally equivalent to the single fault closest to the primary output on that path, we will assume that each multiple fault under consideration is such that it has at the most one single fault on each such path. It should be clear that these two assumptions do not in any way reduce the number of effective multiple faults under consideration. The implications of these two assumptions will be soon made apparent.

B. Notational Preliminaries

To introduce the notational preliminaries used in the paper, consider again the internal fan-out-free network, N, shown in Fig. 1(b). Let $K = \{2, 4, 6, 8\}$ be a set of 4 lines in N. A simultaneous multiple stuck-type fault involving all the four lines of K will be denoted as a binary fault vector $\alpha_K^i = (\alpha_2, \alpha_4, \alpha_6, \alpha_8)$, where $\alpha_t \epsilon \{0, 1\}$, $t \epsilon K$, and i is an index parameter. Clearly, there can be 2^4 different such fault vectors on K. The set of these 2^4 fault vectors on K will be denoted by $F_K = \{\alpha_K^1, \alpha_K^2, \cdots, \alpha_K^{2^4}\}$ and referred to as the *fault complex* on K. Whenever possible, we will write α_K instead of α_K^i to denote a general element of F_K.

Next, let $Y_2 = x_2$, $Y_4 = x_3$, $Y_6 = x_5$, and $Y_8 = x_7$ be the Boolean functions realized by lines 2, 4, 6, and 8, respectively. These Y's will be referred to as *line variables* to differentiate them from the primary input variables. The primary output function of N in terms of the primary input variables, $x_1, x_2, \cdots, x_7$, and the line variables, Y_2, Y_4, Y_6, Y_8, will be written as $Z[X; Y_K]$, where

$$Z[X; Y_K] = (\bar{x}_1 Y_2 + x_2 Y_4)(x_4 Y_6 + \bar{x}_6 Y_8). \quad (2.1)$$

$Y_K = \{Y_2, Y_4, Y_6, Y_8\}$ and $X = \{x_1, x_2, \cdots, x_7\}$. Given a specific input vector, denoted as X_i, the Boolean output of N will then be $Z[X_i; Y_K]$ when N is fault-free, and $Z[X_i; \alpha_K]$ when a $\alpha_K \in F_K$ exists in N.

Note that each $Y_t \in Y_K$ appears exactly once in $Z[X; Y_K]$. In fact, because of the two assumptions made earlier about the multiple faults in internal fan-out-free networks, it should be clear that any set K of lines we consider will always be such that $Z[X; Y_K]$ can be expressed in a form wherein each $Y_t \in Y_K$ occurs exactly once in the expression, either as Y_t or $\bar{Y}_t$. Such a form is referred to as the fan-out-free form [3]. In the following each $Z[X; Y_K]$ will always be assumed to have been expressed in the fan-out-free form.

Definition 1: Two sets, say, I and J are said to form a partition of K if I and J are disjoint sets of lines in N such that $I \cup J = K$ and there exist Boolean functions, say, $H[X; Y_I]$, $M[X; Y_J]$, and $O[X]$ so that $Z[X; Y_K] = (H[X; Y_I] * M[X; Y_J]) + O[X]$, where $*$ denotes Boolean OR, "+", or Boolean AND, "·".

Note that because of the fan-out-free form of $Z[X; Y_K]$, each K will always have at least one partition. For instance, in our previous example, $I = \{2, 4\}$ and $J = \{6, 8\}$ form a partition of K with $H[X; Y_I] = (\bar{x}_1 Y_2 + x_2 Y_4)$, $M[X; Y_J] = (x_4 Y_6 + \bar{x}_6 Y_8)$, and $O[X] = 0$. Using this concept of partition in a recursive manner, it is possible to define an *ordering* on any set of lines in N. More particularly, suppose that sets I and J were some ordered sets $I = (i_1, i_2, \cdots, i_p)$ and $J = (j_1, j_2, \cdots, j_q)$. Then the set K of which I and J form a partition would be ordered as $K = (i_1, i_2, \cdots, i_p, j_1, j_2, \cdots, j_q)$. With this ordering on K, it is possible to write $Z[X; Y_K]$ such that each $Y_t \in Y_K$ appears in $Z[X; Y_K]$ in the same order (from left to right) as the corresponding t appears in K. For a given K, such an ordered $Z[X; Y_K]$ will be unique except for variations possible

because of the associative and commutative properties of "+" and "·".

Another advantage of the ordering scheme is that the fault complex on K can be very conveniently expressed in terms of fault complexes on its partition sets. In particular note that if I, J, and K were ordered as mentioned above, then the Cartesian product $F_I \times F_J$ will be isomorphic to F_K. Thus, for notational convenience we will write that $F_K = F_I \times F_J$ whenever I and J form a partition of K. More generally, if $A \subseteq F_I$ and $B \subseteq F_J$, $A \times B$ will represent a subset of F_K.

C. Motivation

Let T_s be an SFDTS for N, K be a set of lines in N, and suppose that we want to determine all the fault vectors of F_K which are covered by T_s. Since a $\alpha_K \in F_K$ is covered by an X_i iff $Z[X_i; Y_K] \neq Z[X_i; \alpha_K]$, it is clear that any method used to solve this problem would directly or indirectly need all the information contained in $Z[X; Y_K]$. However, because $Z[X; Y_K]$ is inherently dependent on the particular K and N under consideration, it is equally obvious that all such methods are bound to be impractical for any general use. Thus our motivation is to develop a method which will utilize the minimal possible information about K, N, and T_s, and still be able to determine some reasonable information about the capability of T_s to cover F_K. It will be shown in this paper that it is possible to determine the greatest lower bound on the number of fault vectors of F_K covered by any SFDTS for N simply by a table look-up process. In other words, the information used about K and N will be such that it can be tabularized for general use. However, as a consequence of not using complete information about $Z[X; Y_K]$, we will get only greatest lower bounds on coverage.

The main concept developed to obtain our results is that of the *generic representation of logical interconnection of line variables*. To first illustrate what is meant by the logical interconnection of line variables, consider again the set $K = \{2, 4, 6, 8\}$ of lines in N of Fig. 1(b), wherein $Z[X; Y_K] = (\bar{x}_1 Y_2 + x_2 Y_4)(x_4 Y_6 + \bar{x}_6 Y_8)$. Given any two line variables, such as Y_2 and Y_4, when we refer to their logical interconnection we mean the logical AND or logical OR operation which occurs between them in the fan-out-free form of $Z[X; Y_K]$. Thus, Y_2 and Y_4 are logically ORed, and so are Y_6 and Y_8. But Y_2 and Y_6 are logically ANDed, as are Y_4 and Y_6, Y_2 and Y_8, and Y_4 and Y_8. This describes the complete logical interconnection of all the line variables for the given K, and can be succinctly restated as follows: Y_2 and Y_4 are logically ORed and the two together are logically ANDed with the logically ORed Y_6 and Y_8. As another example, let $Z[X; Y_K] = (x_1 Y_2 + \bar{x}_3 x_4 \bar{Y}_3)$ for $K = \{2, 3\}$ in some N. Note that even though $Z[X; Y_K]$ is written in the fan-out-free form, the logical interconnection between Y_2 and Y_3 is that of NOR, which is not included in our definition. By using the Demorgan's Law, however, $Z[X; Y_K]$ can be rewritten as $(\bar{x}_1 + \bar{Y}_2)(x_3 + \bar{x}_4 + Y_3)$, indicating that the logical interconnection between Y_2 and Y_3 is that of AND. Thus, to use our concept it will be assumed in the following that $Z[X; Y_K]$ is always written such that there is no complementation operation involving two or more line variables. Also note from the above two examples that this concept of logical interconnection of line variables is not concerned with whether or not a line variable is complemented, or whether primary input variables are ANDed or ORed with a line variable.

Given the information on the logical interconnection of line variables for a set K in N, by the generic representation of this information we mean a representation in which no information is kept of the particular line variables involved but only of how these line variables are logically interconnected. Thus, the generic representation in the case of $Z[X; Y_K] = (\bar{x}_1 Y_2 + x_2 Y_4)(x_4 Y_6 + \bar{x}_6 Y_8)$ is that there are two logically ANDed pairs of two logically ORed line variables each. We will show in the next section that such generic representations of any set of lines in any network can be conveniently expressed in terms of L-expressions.

III. L-Expressions and Generic Representation

A. L-Expressions

An L-expression is an algebraic expression consisting of a given number of "L" symbols which are interconnected by commutative and associative "+" and "·" operators and parentheses. More formally, we define L-expressions as follows:

Definition 2:

a) (L) is an L-expression.

b) If (P) and (Q) are L-expressions, so are $((P) + (Q))$ and $((P) \cdot (Q))$

c) Nothing else is an L-expression unless its being so follows from a repeated use of a) and b).

Two L-expressions are said to be "equal" or "same" if one can be made identical to the other by using commutative and associative properties of "+" and "·", namely,

$$(P) + (Q) = (Q) + (P)$$
$$(P) \cdot (Q) = (Q) \cdot (P)$$
$$(P) + ((Q) + (R)) = ((P) + (Q)) + (R)$$
$$(P) \cdot ((Q) \cdot (R)) = ((P) \cdot (Q)) \cdot (R).$$

Thus, for example, $((L) + (L)) \cdot (L)$ is the same as $(L) \cdot ((L) + (L))$. For the sake of convenience, the use of parentheses in writing L-expressions will be minimized whenever there is no confusion. Moreover, we will replace all the "·" sign appearances by simple concatenations. Using these conventions, all L-expressions consisting of 4 and less "L" symbols are shown in Table I.

Let $L(v)$ denote the set of all L-expressions each of which consists of v "L" symbols. Each element of $L(v)$ will be referred to as an L-expression of size v, and denoted as L_v^i, $i = 1, 2, \cdots, |L(v)|$, or simply as L_v whenever possible. Note now from Table I and Definition 2 that for all $v > 1$, there is an inherent duality among all L-expressions of $L(v)$. More particularly, we have that if $L_v = L_p + L_q$ (or, $L_v = L_p L_q$), for some $L_p \in L(p)$ and $L_q \in L(q)$, $p + q = v$, then there exists the dual L-expression, denoted L_v^d, of L_v in $L(v)$ such that $L_v^d = L_p^d L_q^d$ (correspondingly, $L_v^d = L_p^d + L_q^d$), where of course $L_p^d \in L(p)$ and $L_q^d \in L(q)$ are the dual L-expressions of L_p and L_q, respectively. An L-expression L_v will be said to be essentially-AND (essentially-OR) if $L_v = L_p L_q$ ($L_v = L_p + L_q$) for some L_p and L_q. Clearly, each L-expression is either essentially-AND

TABLE I

v	Essentially - AND	Essentially - OR
1	L	L
2	LL	L+L
3	LLL	L+L+L, L+LL
4	LLLL, LLL+L,	L+L+L+L, L+L+LL
	(L+L)(L+L), (L+LL)L	LLL+L, LL+L)+L
	(L+L)(L+L)	LL+LL

TABLE II

v	|L(v)|
1	1
2	2
3	4
4	10
5	24
6	66
7	180
8	522
9	1532
10	4624

or essentially-OR but not both, and the dual of an essentially-AND (essentially-OR) L-expression is an essentially-OR (essentially-AND) L-expression.

We end this subsection with the following two lemmas about the cardinality of the set $L(v)$. For more details about the enumeration of different L-expressions the reader is referred to [17] and [9] wherein it is shown that the enumeration of $|L(v)|$ is the same as the enumeration of two-terminal series-parallel networks [18]-[20]. $|L(v)|$, for all $v \leq 10$, is listed in Table II.

Lemma 1: Given any set $L(v)$, $v \geq 2$, there are exactly $|L(v-1)|$ essentially-AND (essentially OR) L-expressions in $L(v)$ each of which can be written as $L_{v-1}L$ (correspondingly, as $L_{v-1} + L$), where $L_{v-1} \epsilon L(v-1)$.

The proof of this lemma follows from Definition 2.

Lemma 2: For large v, $|L(v)| \simeq (0.43) \times (3.56)^v \times v^{-3/2}$. Thus, $|L(v-1)| \simeq 0.28\,|L(v)|$.

The proof of this lemma can be found in [19].

B. Generic Representation

Consider $Z[X; Y_K] = (\bar{x}_1 Y_2 + x_2 Y_4)(x_4 Y_6 + \bar{x}_6 Y_8)$ for $K = \{2, 4, 6, 8\}$ in Fig. 1(b). From our previous discussions, we know that the information about the generic representation of the logical interconnection of line variables of Y_K in this case is that there are two logically-ANDed pairs of two logically ORed line variables each. This information is unambiguously and conveniently represented by the L-expression $(L + L)(L + L)$ if each "L" is assumed to stand for a line variable. More particularly, given any set K of lines in N, we can obtain by using the following procedure an L-expression from the corresponding $Z[X; Y_K]$ such that the resulting L-expression will generically represent the logical interconnection of line variables in $Z[X; Y_K]$.

Procedure: Given a $Z[X; Y_K]$ in the fan-out-free form, $|K| = v$, first replace each literal of X in $Z[X; Y_K]$ by a "0" or "1" such that the simplified expression consists only of all the v line variables. Then, replace each line variable in the simplified expression by "L". The resulting expression is the required L-expression, $L_v \in L(v)$.

It is obvious from this procedure and the fact that each $Z[X; Y_K]$ can be uniquely written in an ordered, fan-out-free form that the resulting L-expression will be unique for a given set of lines. Such an L-expression will be said to be the *associated* L-expression with K.

In the remainder of this paper we will show that given a set K of v lines in any N and its associated L-expression, L_v, it is possible to determine the greatest lower bound on the number of fault vectors of F_K covered by every SFDTS for N simply by knowing L_v. An immediate corollary of this result is that for all possible sets of v lines, in all possible internal fan-out-free networks, with which the associated L-expression is the same, the greatest lower bound on the coverage will also be the same. Thus, if the greatest lower bound corresponding to each possible L-expression were recorded in a table, then given any set K of lines in any N, the bound may be obtained simply by a table look-up process. This general and simple use of L-expressions will be the result developed and validated in the following sections of the paper.

IV. Coverage Table

A. The Concept of a Coverage Table

Consider an internal fan-out-free network N, and a set K of v lines in N. Let T be any set of input vectors which detects all $2v$ single stuck-type faults on K. (T, of course, could also detect other faults.) Relative to N, let $\phi(K,T)$ denote the number of fault vectors of F_K also detected by T. Finally $\phi(K)$ be the minimum $\phi(K,T)$ for all such T in N. Since each SFDTS for N automatically covers all $2v$ single faults on K, it follows that at least $\phi(K)$ fault vectors of F_K must be covered by each SFDTS for N.

Suppose now that the L-expression associated with K is L_v. In Section IV-B an algorithm is presented which, given an L-expression such as L_v, determines a unique number, denoted as $\theta(L_v)$. In the next section we will prove that $\theta(L_v)$ is the greatest lower bound on $\phi(K)$ in the following sense:

Property P1: Given any N and a set K of v lines with which L_v is the associated L-expression,

$$\theta(L_v) \leq \phi(K).$$

Property P2: There exists at least one N with a set K of v lines with which L_v is the associated L-expression and

$$\theta(L_v) = \phi(K).$$

In other words, each SFDTS of N covers at least $\theta(L_v)$ fault vectors of F_K, and there exists at least one N and an SFDTS for it which covers no more than $\theta(L_v)$ fault vectors of F_K. Thus, $\theta(L_v)$ provides the greatest lower bound on the capability of any SFDTS to cover fault vectors of F_K. Note that $\theta(L_v)$ is the greatest lower bound over all possible sets of lines in all possible networks with which the associated L-expression is L_v. Thus, it might happen in a general network that each SFDTS for the network covers more than $\theta(L_v)$ fault vectors of each set of lines with the associated L-expression, L_v. Regardless, our main interest lies only in the most general nature of multiple fault coverage capabilities of single fault detection test sets.

The table in which all L-expressions and their corresponding

TABLE III

v	L_v^i	$\theta(L_v^i)$	v	L_v^i	$\theta(L_v^i)$
1	L	2	6	(LLL+L) (L+L)	56
2	L+L	4		((L+L) L+L) (L+L)	56
	LL	4		(LL+L+L) (L+L)	54
3	L+L+L	8		(L+L+L+L) (L+L)	56
	LL+L	8	7	((LL+L) L+L) (L+L)	112
	(L+L) L	8		((L+L+L) L+L) (L+L)	112
	LLL	8		((L+L) (L+L)+L) (L+L)	104
4	L+L+L+L	16		((L+L) LL+L) (L+L)	112
	LL+L+L	16		(LLLL+L) (L+L)	112
	(L+L) L+L	16		((L+L) L+LL) (L+L)	104
	LLL+L	16		(LLL+LL) (L+L)	106
	(L+L+L) L	16		((L+L) L+L+L) (L+L)	112
	(LL+L) L	16		(LLL+L+L) (L+L)	112
	(L+L) LL	16		(LL+LL+L) (L+L)	104
	LLLL	16		(LL+L+L+L) (L+L)	106
	(L+L) (L+L)	14		(L+L+L+L+L) (L+L)	112
	LL+LL	14		((L+L) L+L) (LL+L)	106
5	(LL+L) (L+L)	28		((L+L) L+L) (L+L+L)	104
	(L+L+L) (L+L)	28		(LLL+L) (LL+L)	106
6	(LL+L) (LL+L)	54		(LLL+L) (L+L+L)	106
	(LL+L) (L+L+L)	54		(LL+LL) (LL+L)	106
	(L+L+L) (L+L+L)	52		(LL+LL) (L+L+L)	102
	(L+L) (L+L) (L+L)	52		(LL+L+L) (LL+L)	100
	(LL+LL) (L+L)	56		(LL+L+L) (L+L+L)	102
				(L+L+L+L) (LL+L)	106
				(L+L+L+L) (L+L+L)	106
				(LL+L) (L+L) (L+L)	106
				(L+L+L) (L+L) (L+L)	102

$\theta(L_v)$'s are listed will be referred to as the Coverage Table. Table III shows the Coverage Table in a compact form (see Section IV-C) for all L-expressions of size 7 and less. Various ramifications of this table are considered later in the paper.

B. $\theta(L_v)$ Calculation

To determine $\theta(L_v)$, we give the following algorithm in which the reader's familiarity with the cubical complex notation [21], [22] is assumed.

Coverage Algorithm:

Input: L_v, an L-expression consisting of v L's.

Output: $\theta(L_v)$, the greatest lower bound on the minimum coverage.

Comments: This algorithm uses two sets of v-dimensional cubes denoted, respectively, as $D^0(L_v)$ and $D^1(L_v)$. These sets are determined by a recursion. This recursion will at times utilize the r-dimensional unit cube $S_r = (\underbrace{xx \cdots x}_{r})$, $r \geq 1$. Two families of subsets of $D^0(L_v)$ and $D^1(L_v)$ are next found, and magnitudes of two sets of representatives (SR's) for these families finally provide the required $\theta(L_v)$. A set of representatives of a family of sets of cubes, say $\{C^1, C^2, \cdots, C^n\}$ is defined as a set of cubes $C = \{c_1, c_2, \cdots, c_r\}$ such that for each set C^i, $1 \leq i \leq n$, there exists at least one element, say, c_j in C which also belongs to C^i.

Method:

Step 1: If $v = 1$, set $D^0(L_v) = (0)$ and $D^1(L_v) = (1)$. Go to Step 4.

Step 2: if $v \geq 2$, let L_p and L_q, $p + q = v$, be such that $L_v = L_p + L_q$ or $L_v = L_pL_q$. Furthermore, assume that $D^0(L_p)$, $D^1(L_p)$, $D^0(L_q)$, and $D^1(L_q)$ are known by previous recursive use of this algorithm.

Step 3(a): If $L_v = L_p + L_q$, set $D^0(L_v) = D^0(L_p) \times D^0(L_q)$ and $D^1(L_v) = (D^1(L_p) \times S_q) \cup (S_p \times D^1(L_q))$.

Step 3(b): If $L_v = L_pL_q$, set $D^0(L_v) = (D^0(L_p) \times S_q) \cup (S_p \times D^0(L_q))$ and $D^1(L_v) = D^1(L_p) \times D^1(L_q)$.

Step 4: Let $D_t^0(L_v)$ be the set of all those cubes of $D^0(L_v)$ each of which has a "0" in the tth coordinate position, where $t = 1, 2, \cdots, v$. Similarly, let $D_t^1(L_v)$ be the set of all those cubes of $D^1(L_v)$ each of which has a 1 in the tth coordinate position. Moreover, let $P^0(L_v)$ denote an SR of the family $\{D_1^0(L_v), D_2^0(L_v), \cdots, D_v^0(L_v)\}$ and $P^1(L_v)$ an SR of the family $\{D_1^1(L_v), D_2^1(L_v), \cdots, D_v^1(L_v)\}$. Finally, let $||C||$ denote the number of 0-cubes in a set, C, of cubes.

Step 5: Let $P^0_{\min}(L_v)$ be an SR such that $||P^0_{\min}(L_v)|| \leq ||P^0(L_v)||$ for every SR $P^0(L_v)$. Similarly, let $P^1_{\min}(L_v)$ be an SR such that $||P^1_{\min}(L_v)|| \leq ||P^1(L_v)||$ for every SR $P^1(L_v)$.

Step 6: $\theta(L_v) = ||P^0_{\min}(L_v)|| + ||P^1_{\min}(L_v)||$.

Stop.

The following example illustrates the algorithm.

Consider $L_v = (L + L)(L + L)$, $v = 4$. Letting $L_p = L + L$ and $L_q = L + L$, we first determine $D^0(L + L)$ and $D^1(L + L)$ and then use these sets to determine $D^0(L_v)$ and $D^1(L_v)$. Thus, from Step 3(a)

$$D^0(L + L) = D^0(L) \times D^0(L) = (00)$$

and

$$D^1(L + L) = (D^1(L) \times S_1) \cup (S_1 \times D^1(L) = (1x, x1).$$

Now using Step 3(b) next, we have

$$D^0(L_v) = (D^0(L+L) \times S_2) \cup (S_2 \times D^0(L+L))$$
$$= (00xx, xx00)$$

and

$$D^1(L_v) = D^1(L+L) \times D^1(L+L)$$
$$= (1x1x, 1xx1, x11x, x1x1).$$

Moving on to Step 4 yields

$$D_1^0(L_v) = D_2^0(L_v) = (00xx)$$
$$D_3^0(L_v) = D_4^0(L_v) = (xx00)$$
$$D_1^1(L_v) = (1x1x, 1xx1)$$
$$D_2^1(L_v) = (x11x, x1x1)$$
$$D_3^1(L_v) = (1x1x, x11x)$$

and

$$D_4^1(L_v) = (1xx1, x1x1).$$

Note that there is only one SR of the family $|D_1^0(L_v), \cdots, D_4^0(L_v)|$ namely, $P_{\min}^0(L_v) = (00xx, xx00)$. However, there are various SR's of the second family. Examining the possibilities shows that the two candidates for $P_{\min}^1(L_v)$ are $(1x1x, x1x1)$ and $(x11x, 1xx1)$. Thus, we finally get that

$$\theta(L_v) = ||P_{\min}^0(L_v)|| + ||P_{\min}^1(L_v)||$$
$$= 7 + 7$$
$$= 14.$$

From this example (and anticipating our proofs in the following section that $\theta(L_v)$ so calculated does, indeed, have Properties P1 and P2), we can conclude here that for any set of 4 lines which are generically represented by $(L+L)(L+L)$ in any internal fan-out-free network, any set of input vectors which detects all of the 8 single faults on these lines also detects at least 14 of the 16 multiple faults involving all 4 of these lines.

It should be cited that the most difficult aspect of the implementation of the Coverage Algorithm is clearly the exhaustive step of determining $P_{min}^0(L_v)$ and $P_{min}^1(L_v)$. However, since each $\theta(L_v)$ entry in the Coverage Table must be calculated only once, such complexity is amortized by its general use. Moreover, this situation is alleviated somewhat by the consideration of the following subsection.

C. Compacting the Coverage Table

Note from Table II that there are 24, 66, and 180 L-expressions of sizes, 5, 6, and 7, respectively. Yet, for these three sizes only 2, 9, and 24 L-expressions, respectively, have been listed in Table III. This is a result of the following two lemmas, the proofs of which are omitted here but are directly based on the Coverage Algorithm [17].[1]

Lemma 3: If $L_v = L_{v-1}L$ (or $L_v = L_{v-1} + L$), then

$$\theta(L_v) = \theta(L_{v-1}) + 2^{v-1}.$$

Lemma 4: $\theta(L_v) = \theta(L_v^d)$, where L_v^d is the dual L-expression of L_v.

From Lemmas 1 and 3 it follows that for each set $L(v)$, it is sufficient to calculate $\theta(L_v)$, $L_v \epsilon L(v)$, for only $|L(v)| - 2|L(v-1)|$ L-expressions of size v. Moreover, from Lemma 4 it follows that $\theta(L_v)$ must actually be calculated for only one-half of these $|L(v)| - 2|L(v-1)|$ L-expressions. In other words, it is sufficient to list in the table entries for only $\frac{1}{2}|L(v)| - |L(v-1)|$ of the L-expressions of size v. By Lemma 2, this means we need to explicitly list only 22 percent of all the entries for $L(v)$.

[1] For illustrative purposes Lemmas 3 and 4 have not been used in compacting Table III for L-expressions of size 4 and less.

D. Applications of the Coverage Table

The most straightforward use of the Coverage Table is, as pointed out before, to determine the greatest lower bound, $\theta(L_v)$, on the number of fault vectors of F_K covered by any SFDTS in any N for any set K of lines with which the associated L-expression is L_v. Thus, for example, note from Table III that since $\theta(L_v) = 2^v = |F_K|$ for all L_v, $v \leq 3$, we can state the following.

Lemma 5: Given any SFDTS, say, T_s for any internal fan-out-free network, N, T_s covers all the multiple faults of sizes 2 and 3 as well.

This, of course, is a previously known result [10]–[12]; however, the generality of the Coverage Table approach gives this result a new perspective. To illustrate further, note from Table III that any SFDTS for N will also cover all the multiple faults of size 4 except for the cases where the L-expression associated with a set K of 4 lines is either $LL + LL$ or the dual $(L+L)(L+L)$. Moreover, even in these two cases, it can be seen from the table that at least 14 out of 16 fault vectors of F_K will necessarily be covered by each SFDTS. A similar observation about L-expressions of size 5 together with the above observation leads to the following:

Lemma 6: Given any SFDTS, say, T_s for any internal fan-out-free N, T_s covers at least 14 out of 16 fault vectors on any set of 4 lines and at least 28 out of 32 fault vectors on any set of 5 lines in N.

Similar observations could, of course, be made regarding multiple faults of larger sizes.

Another interesting type of use of the Coverage Table is to perform what we will call coverage averaging. This involves the use of the Coverage Table to determine the average capability of an SFDTS to cover all the multiple faults in a given network of some predetermined size. To do this we make the assumption that each L-expression $L \epsilon L(v)$ is equinumerous in N. We can then calculate

$$\text{AVG}[\theta(L_v)] = \frac{\sum_{L_v \epsilon L(v)} \theta(L_v)}{2^v |L(v)|}.$$

This quantity represents the average value of $\theta(L_v)$ over all $\theta(L_v)$'s of fixed size v, divided by 2^v. With an equinumerous assumption, we can then conclude that in any network N, on the average $\text{AVG}[\theta(L_v)]$ of all multiple faults of size v will necessarily be covered by every SFDTS for N. A graph which shows the values of $\text{AVG}[\theta(L_v)]$ for all $v \leq 8$ is given in Fig. 2. The following theorem is a direct conclusion from this figure.

Theorem 1: On the average, an SFDTS in an internal fan-out-free network covers 92 percent of all multiple faults of sizes 8 or less.

Clearly, the validity of this theorem depends on the validity of the equinumerous assumption mentioned above; however,

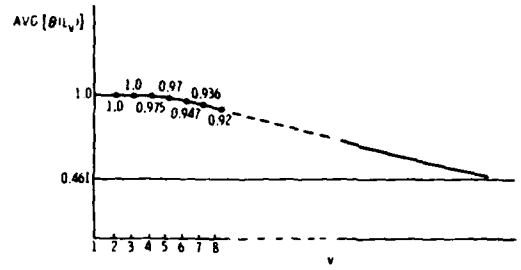


Fig. 2. AVG[$\theta(L_v)$] for $v \leq 8$.

consideration of large general internal fan-out-free N indicates that the assumption is not unduly restrictive and that for many networks of interest this theorem is quite applicable. Moreover, if the equinumerous assumption were not valid for a network, a weighted coverage averaging could be performed.

Now, this theorem together with some recent results by Goldstein [16] represent a quantitative affirmation of the common speculation that, at least in internal fan-out-free networks, SFDTS's cover most multiple faults as well. More specifically, Goldstein [16] has shown by exploiting statistical information relating to physical defects and chip layout data that for most networks consideration of multiple faults up to size 8 is sufficient for multiple fault analysis. Of course, for networks where the consideration of at most 8 faults is not sufficient, Theorem 1 could be accordingly extended.

A very useful result obtained by coverage averaging under the equinumerous assumption for large networks is the following theorem.

Theorem 2: For all $v \geq 1$, AVG[$\theta(L_v)$] > 0.461

A proof of this theorem using the asymptotic nature of $\theta(L_v)$ is derived in [17]. The implication of the theorem however is that on the average at least 46.1 percent of all multiple faults are covered by each SFDTS in an internal fan-out-free network. This is the first quantitative result which provides an estimate of the capability of SFDTS's to cover all multiple faults in networks which are more general than restricted fan-out-free networks [13], the largest class of networks for which it is known that every SFDTS covers all multiple faults.

V. Justification of $\theta(L_v)$

In this section Properties P1 and P2 of $\theta(L_v)$ which were described in the previous section and which justify our associated claims regarding $\theta(L_v)$ will be considered. P1 will be proved in Section V-A and a procedure to construct a network for any L_v such that P2 holds will be given in Section V-B. In the following, our arguments for convenience and with no loss of generality will be developed under the assumptions that the set K of lines being considered is such that each $Y_i \epsilon Y_K$ appears in $Z[X; Y_K]$ in the uncomplemented form only, and that the set $K = \{1, 2, \cdots, v\}$.

We start with the following two definitions which were introduced in [10].

Definition 3: Given a set K of lines in N and an input vector X_i to N, a fault vector $\alpha_K \epsilon F_K$ will be said to be a *$(0 \rightarrow 1)$-tested fault vector for X_i* if $Z[X_i; Y_K] = 0$ and $Z[X_i; \alpha_K] = 1$. Similarly, a fault vector $\alpha_K \epsilon F_K$ will be said to be a $(1 \rightarrow 0)$-tested fault vector for X_i if $Z[X_i; Y_K] = 1$ and $Z[X_i; \alpha_K] = 0$.

Definition 4: Given an input vector X_i to N, a fault vector $\alpha_K \epsilon F_K$ will be said to be a β-consistent fault vector for X_i relative[2] to $A[X; Y_K]$ if $A[X_i; Y_K] = A[X_i; \alpha_K] = \beta$, $\beta \epsilon \{0, 1\}$.

Clearly, a β-consistent fault vector is such that the Boolean value of $A[X, Y_K]$ for X_i is not changed by the presence of α_K. For example, if $A[X; Y_K] = x_1Y_1 + x_2Y_2$ for $K = \{1, 2\}$, and $Y_1 = x_3$ and $Y_2 = x_4$, then $\alpha_K = (0, 1)$ is 0-consistent for $X_i = (1001)$. Similarly, the fault $(0, 0)$ is also seen to be 0-consistent for X_i.

It is shown in [10] that the concepts of Definitions 3 and 4 are of fundamental importance in determining the *exact* coverage of fault vectors of F_K by input vectors which are known to cover fault vectors of F_I and F_J, where I and J form a partition of K. In this paper, we use the same concepts instead to obtain the greatest lower bound on the exact coverage. The main difference between the two approaches is that whereas in [10] the complete functional information contained in $Z[X; Y_K]$ is used to calculate the exact coverage, we use in this paper only the information contained in the associated L-expression to calculate the bound. To illustrate the difference, consider a set of K of lines such that $Z[X; Y_K] = H[X; Y_I] + M[X; Y_J] + O[X]$. Let $\alpha_I \epsilon F_I$ be a $(1 \rightarrow 0)$-tested fault vector for some X_i. In other words, $Z[X_i; Y_I] = 1$ and $Z[X_i; \alpha_I] = 0$. Since $Z[X; Y_I]$ is the Boolean function realized by N, it is the same as $Z[X; Y_K] = H[X; Y_I] + M[X; Y_J] + O[X]$ if all $Y_i \epsilon Y_J$ in $M[X; Y_J]$ are replaced by their equivalent primary input Boolean functions. Thus, it follows that

$$H[X_i; Y_I] + M[X_i; Y_J] + O[X_i] = 1$$

and

$$H[X_i; \alpha_I] + M[X_i; Y_J] + O[X_i] = 0.$$

That is, $H[X_i; Y_I] = 1$, $H[X_i; \alpha_I] = 0$, $M[X_i; Y_J] = 0$, and $O[X_i] = 0$.

Next, let $\alpha_J \epsilon F_J$ be a 0-consistent fault vector for X_i relative to $M[X_i; Y_J]$; that is, $M[X_i; Y_J] = M[X_i; \alpha_J] = 0$. It is then easily seen that $\alpha_K = (\alpha_I, \alpha_J)$ is also a $(1 \rightarrow 0)$-tested fault vector for X_i. In other words, we have that if α_I is a $(1 \rightarrow 0)$-tested fault vector for X_i, α_J is any 0-consistent fault vector for X_i, and $Z[X; Y_K] = H[X; Y_I] + M[X; Y_J] + O[X]$, then $\alpha_K = (\alpha_I, \alpha_J)$ is also a $(1 \rightarrow 0)$-tested fault vector for X_i. More specifically, if we define

$$G_K^{10}(X_i) = \{\alpha_K \mid Z[X_i; Y_K] = 1 \text{ and } Z[X_i; \alpha_K] = 0\}$$

and

$$F_J^0(X_i) = \{\alpha_J \mid M[X_i; Y_J] = M[X_i; \alpha_J] = 0\},$$

then for the above example we have that

$$G_K^{10}(X_i) = G_I^{10}(X_i) \times F_J^0(X_i). \tag{5.1}$$

By recursively using relations such as (5.1) above, it is

[2] $A[X; Y_K]$ is used to denote a general Boolean expression in the fan-out-free form in terms of the variables of X and Y_K. In particular, $A[X; Y_K]$ could be the same as $Z[X; Y_K]$.

possible [10] to determine the exact coverage of F_K by an SFDTS. This approach, however, is not very practical because sets such as $F_J^0(X_i)$ are highly dependent on the particular X_i, J, and $M[X; Y_J]$ under consideration.

It turns out though, as will be shown in the following, that it is possible to determine a subset of $F_J^0(X_i)$ simply by knowing the L-expression associated with J. That is, the generic information of logical interconnection of line variables of Y_J in $M[X; Y_J]$ can be suitably exploited to obtain a subset of $F_J^0(X_i)$. Since such generic information, namely the associated L-expression with J, is independent of the particular X_i, J, and $M[X; Y_J]$ under consideration, such subsets can clearly be used in infinitely many different situations.

Furthermore, it is also possible to show that a subset of $G_I^{10}(X_i)$ can be obtained simply by knowing the L-expression associated with I. Using these two results, it then follows from (5.1) that a subset of $G_K^{10}(X_i)$ can be obtained by using the L-expression associated with K. In fact, the Coverage Algorithm of the previous section does exactly this, as we shall prove in the following two lemmas. It should now be clear, however, that the greatest lower bound on the exact coverage of F_K is directly related to such subsets of $G_K^{01}(X_i)$ and $G_K^{10}(X_i)$, $X_i \epsilon T_s$. We will consider more details of this in Sections V-A and V-B.

Before presenting the two lemmas, it will be convenient to introduce some notation here. Because of the generic approach we are taking, we will represent fault vectors and fault complexes in the following by using the same cubical complex notation as is used in the Coverage Algorithm. In this notation, a fault vector, say, α_K will be considered as a v-dimensional 0-cube. The fault complex F_K will then simply be the v-dimensional unit cube $(xx \cdots x)$. Moreover, a subset of F_K is then represented as a set of various cubes in the v-dimensional unit space. The use of these notations will be clearer as we proceed.

Lemma 7: Let $K = \{1, 2, \cdots, v\}$ be any set of v lines in N and let $A[X; Y_K]$ be a fan-out-free form Boolean expression. Moreover, let L_v be the L-expression associated with K relative to $A[X; Y_K]$. Then given any X_i, $A[X_i; Y_K] = \beta$, there exists at least one cube, say, $c \epsilon D^\beta(L_v)$ such that the set of 0-cubes of c is contained in or equal to $F_K^\beta(X_i)$, $\beta \epsilon \{0, 1\}$.

Before we enter into a formal proof, the scope of this lemma should be emphasized again. Recall that in the Coverage Algorithm, the set of cubes $D^\beta(L_v)$ is determined with no reference to a particular network N, or a particular set K of lines, or a particular input vector X_i. Indeed, we only considered a set of v lines in any N whose logical interconnection was generally represented by L_v. Yet, in spite of the generality under which $D^\beta(L_v)$ is generated, we nevertheless are stating in Lemma 7 that this set of cubes contains a subset of $F_K^\beta(X_i)$ where, by definition, in this latter set we are referring to particular N, K, and X_i—and this result holds regardless of the particulars. That is, it holds for each of the infinite number of cases in which L_v can be associated with some set of lines in an internal fan-out-free network. Hence, Lemma 7 makes a fundamental connection between the generic and the specific which is at the very center of all our results.

Proof: Let $v = 1$. Then $K = \{1\}$ and $A[X; Y_K] = A_1[X]Y_1 + A_2[X]$ for some $A_1[X]$ and $A_2[X]$. Given now that $A[X_i; Y_K] = A_1[X_i]Y_1 + A_2[X_i] = 0$ (or 1) for some X_i, it immediately follows that $A_1[X_i]\alpha_1 + A_2[X_i] = 0$ (1) for $\alpha_1 = 0$ (1). That is, $F_K^\beta(X_i) \supseteq \{\beta\}$. Recalling now from the Coverage Algorithm that $D^\beta(L_v) = (\beta)$ for $v = 1$, we have that Lemma 7 is true for $v = 1$.

Assume now that the lemma is true for every set K of 1, 2, $\cdots$, and $v-1$ lines, where $v \geq 2$. Based on this assumption, we will now prove that it is also true for every set K of v lines. Thus, let K be a set of v lines and L_v be the L-expression associated with K relative to $A[X; Y_K]$. Let I and J form a partition of K such that $A[X; Y_K] = (A_1[X; Y_I] * A_2[X; Y_J]) + O[X]$. Then, clearly, $L_v = L_p * L_q$, where $L_p(L_q)$ is the L-expression associated with $I(J)$ relative to $A_1[X; Y_I](A_2[X; Y_J])$. We will prove the lemma only for the case where "$*$" is "$+$" and $\beta = 0$. The proof for the remaining cases follows analogously.

Thus, let X_i be such that

$$A[X_i; Y_K] = A_1[X_i; Y_I] + A_2[X_i; Y_J] + O[X_i] = 0.$$

Then clearly,

$$A_1[X_i; Y_I] = A_2[X_i; Y_J] = O[X_i] = 0. \tag{5.2}$$

Now, let α_I be a 0-consistent fault vector for X_i relative to $A_1[X; Y_I]$; that is, let $\alpha_I \epsilon F_I^0(X_i)$. Similarly, let $\alpha_J \epsilon F_J^0(X_i)$. Then by Definition 4,

$$A_1[X_i; \alpha_I] = 0 \text{ and } A_2[X_i; \alpha_J] = 0. \tag{5.3}$$

Using (5.2) and (5.3), it is then easily seen that if $\alpha_K = (\alpha_I, \alpha_J)$, then since

$$A[X_i; \alpha_K] = A_1[X_i; \alpha_I] + A_2[X_i; \alpha_J] + O[X_i] = 0.$$

Therefore α_K must be a 0-consistent fault vector for X_i relative to $A[X_i; Y_K]$. In other words, we have proven that if $\alpha_I \epsilon F_I^0(X_i)$ and $\alpha_J \epsilon F_J^0(X_i)$, then $\alpha_K \epsilon F_K^0(X_i)$. That is,

$$F_I^0(X_i) \times F_J^0(X_i) \subseteq F_K^0(X_i). \tag{5.4}$$

Consider now the sets $D^0(L_p)$ and $D^0(L_q)$. By our induction hypothesis we have that there must exist at least one cube, say, $c_1 \epsilon D^0(L_p)$ and another cube, say $c_2 \epsilon D^0(L_q)$ such that the 0-cubes of c_1 and c_2 are contained in $F_I^0(X_i)$ and $F_J^0(X_i)$, respectively. It then follows from (5.4) that the 0-cubes of the cube $(c_1 c_2)$ will belong to $F_K^0(X_i)$. However, from Step 3(a) of the Coverage Algorithm it is seen that $(c_1 c_2) \epsilon D^0(L_v)$.

Q.E.D.

Lemma 8: Let $K = \{1, 2, \cdots, v\}$ be any set of v lines in N and let L_v be the L-expression associated with K. Moreover, let X_i be an input vector which $(0 \rightarrow 1)$-tests (or $(1 \rightarrow 0)$-tests) the single fault $\alpha_t = 1$ (correspondingly, $\alpha_t = 0$), $t \epsilon K$, on the lines of K. Then there exists at least one cube, say, $c \epsilon D_t^1(L_v)(c \epsilon D_t^0(L_v))$ such that the set of 0-cubes of c is contained in or equal to $G_K^{01}(X_i)(G_K^{10}(X_i))$.

Lemma 8, like Lemma 7, makes a fundamental connection between the generic, $D_t^\beta(L_v)$, $\beta \epsilon \{0, 1\}$, and the specific, $G_K^{01}(X_i)$ and $G_K^{10}(X_i)$.

Proof: First of all note that because of our assumption that each $Y_t \epsilon Y_K$ appears in $Z[X; Y_K]$ in the uncomplemented form, it follows that the s-a-0 fault on any single line $t \epsilon K$ (that is, the fault $\alpha_t = 0$) will always be $(1 \rightarrow 0)$-tested by any X_i

which tests this fault. Similarly, the fault $\alpha_t = 1$ will always be $(0 \rightarrow 1)$-tested by each X_i which tests it. In other words, $G^{10}_{|t|}(X_i) = (0)$ and $G^{01}_{|t|}(X_i) = (1)$ for every $t \epsilon K$.

To prove the lemma now, let $K = \{1\}$ and $v = 1$. Then since $D^1_t(L_v) = (1)$ and $D^0_t(L_v) = (0)$ from the Coverage Algorithm, and since $G^{01}_K(X_i) = (1)$ and $G^{10}_K(X_i) = (0)$ as discussed above, the lemma is clearly true for $v = 1$.

Assume now that the lemma is true for all sets of $1, 2, \cdots, v-1$ lines. Consider then a set $K = \{1, 2, \cdots, v\}$ of v lines. Let I and J form a partition of K, where $I = \{1, 2, \cdots, p\}$ and $J = \{p+1, p+2, \cdots, p+q\}$, $p + q = v$. Let $L_v = L_p * L_q$ be the L-expression associated with K such that $L_p(L_q)$ is the L-expression associated with $I(J)$. We will now prove the lemma for the case wherein

$$Z[X; Y_K] = H[X; Y_I] + M[X; Y_J] + O[X]. \quad (5.6)$$

X_i $(1 \rightarrow 0)$-tests $\alpha_t = 0$, and $t \epsilon I$. The proof for the remaining cases follows in a similar manner. Recall here that for the case mentioned above we have already seen in (5.1) that

$$G^{10}_K(X_i) = G^{10}_I(X_i) \times F^0_J(X_i). \quad (5.7)$$

Moreover, since X_i $(1 \rightarrow 0)$-tests α_t, $t \epsilon I$, it follows from our induction hypothesis that there exists a cube, say, $c^1 \epsilon D^0_t(L_p)$ such that all the 0-cubes of c^1 are contained in $G^{10}_I(X_i)$. Also, from Lemma 7 we have that there exists a cube, say, $c^2 \epsilon D^0(L_q)$ such that all the 0-cubes of c^2 are contained in $F^0_J(X_i)$. Thus it follows from (5.7) that all the 0-cubes of the cube $c = (c^1 c^2)$ will be contained in $G^{10}_K(X_i)$. However, from the Coverage Algorithm we know that $c \epsilon D^0_t(L_v)$ since $D^0_t(L_v) = D^0_t(L_p) \times D^0(L_q)$. Q.E.D.

A. Proof of Property P1

Consider an SFDTS, T_s, for N and let K be any set of v lines in N. The exact capability of T_s to cover fault vectors of F_K can be quantitatively expressed as

$$G_K(T_s) = \bigcup_{X_i \epsilon T_s} G_K(X_i)$$

where $G_K(X_i)$ is $G^{01}_K(X_i)$ or $G^{10}_K(X_i)$ depending on X_i. Clearly, $|G_K(T_s)| = \phi(K, T_s)$, as defined in Section IV. Using Lemmas 7 and 8, we now show in this section that certain subsets of $G_K(X_i)$, $X_i \epsilon T_s$, when combined in the specific manner detailed in the Coverage Algorithm lead to a number, $\theta(L_v)$, which satisfies P1.

Theorem 3: Given a set $K = \{1, 2, \cdots, v\}$ of v lines in N and the minimum coverage $\phi(K)$ of F_K by every SFDTS for N, the integer $\theta(L_v)$ obtained by the Coverage Algorithm is such that

$$\theta(L_v) \leq \phi(K),$$

where L_v is the L-expression associated with K.

Proof: Let T_s be an SFDTS of N such that

$$\phi(K) = \phi(K, T_s) = |G_K(T_s)|. \quad (5.8)$$

Since there are $2v$ single faults on the lines of K, there must exist at least one input vector in T_s corresponding to each of these $2v$ faults. To be more specific, let $X^{01}_t \epsilon T_s$, $t = 1, 2, \cdots, v$, be an input vector which $(0 \rightarrow 1)$-tests the single fault $\alpha_t = 1$; and let $X^{10}_t \epsilon T_s$, $t = 1, 2, \cdots, v$ be an input vector which $(1 \rightarrow 0)$-tests the single fault $\alpha_t = 0$, where, of course, all these vectors might not be distinct vectors. Recalling that

$$\bigcup_{\substack{\text{for all} \\ X_i \epsilon T_s}} G_K(X_i) = G_K(T_s)$$

it is then clear that

$$\bigcup_{\substack{\text{for all} \\ t \epsilon K}} G^{01}_K(X^{01}_t) \bigcup_{\substack{\text{for all} \\ t \epsilon K}} G^{10}_K(X^{10}_t) \subseteq G_K(T_s). \quad (5.9)$$

Consider now Lemma 8. For each $t \epsilon K$, let c^{01}_t be a cube in $D^1_t(L_v)$, all the 0-cubes of which are contained in $G^{01}_K(X^{01}_t)$. Similarly, let c^{10}_t, $t \epsilon K$, be a cube in $D^0_t(L_v)$, all the 0-cubes of which are contained in $G^{10}_K(X^{10}_t)$. Then, it follows from this lemma that

$$\{\text{0-cubes in } \{c^{01}_1, c^{01}_2, \cdots, c^{01}_v\}\} \subseteq \bigcup_{\substack{\text{for all} \\ t \epsilon K}} G^{01}_K(X^{01}_t). \quad (5.10)$$

and

$$\{\text{0-cubes in } \{c^{10}_1, c^{10}_2, \cdots, c^{10}_v\}\} \subseteq \bigcup_{\substack{\text{for all} \\ t \epsilon K}} G^{10}_K(X^{10}_t). \quad (5.11)$$

Clearly, now, $\{c^{01}_1, c^{01}_2, \cdots, c^{01}_v\}$ is an SR, say, $P^{01}(L_v)$ of the family $\{D^1_1(L_v), D^1_2(L_v), \cdots, D^1_v(L_v)\}$, and $\{c^{10}_1, c^{10}_2, \cdots, c^{10}_v\}$ is an SR, say, $P^{10}(L_v)$ of the family $\{D^0_1(L_v), D^0_2(L_v), \cdots, D^0_v(L_v)\}$. Therefore, we have from (5.10) and (5.11) that

$$||P^{01}(L_v)|| \leq |\bigcup_{\substack{\text{for all} \\ t \epsilon K}} G^{01}_K(X^{01}_t)| \quad (5.12)$$

and

$$||P^{10}(L_v)|| \leq |\bigcup_{\substack{\text{for all} \\ t \epsilon K}} G^{10}_K(X^{10}_t)|. \quad (5.13)$$

In other words, using (5.12) and (5.13) in (5.9), we get that

$$||P^{01}(L_v)|| + ||P^{10}(L_v)|| \leq |G_K(T_s)|. \quad (5.14)$$

Finally, recalling from the Coverage Algorithm that

$$\theta(L_v) = ||P^{01}_{\min}(L_v)|| + ||P^{10}_{\min}(L_v)||,$$

where $||P^{01}_{\min}(L_v)|| \leq ||P^{01}(L_v)||$ and $||P^{10}_{\min}(L_v)|| \leq ||P^{10}(L_v)||$ and using (5.8) and (5.14), it is clear that

$$\theta(L_v) \leq \phi(K). \quad \text{Q.E.D.}$$

B. Property P2

Given any L-expression L_v we will in the following first present a procedure which will associate with L_v a specific network, denoted as, $N(L_v)$. A set of lines will next be shown

to exist in $N(L_v)$ such that the L-expression associated with K is L_v. Finally, an SFDTS, T_s, for $N(L_v)$ will be generated with the property that $\phi(K,T_s) = \phi(K) = \theta(L_v)$. For the sake of brevity, no proofs are provided in this section; the interested reader can request [17] for all the pertinent details.

Let L_v be an L-expression consisting of v L's. Then a network of class $C1$, denoted as, $N(L_v)$ which has no primary input fan out and consists of $3v$ primary input variables will be associated with L_v by the procedure given in the following. The network $N(L_v)$ will be constructed by this procedure in a recursive manner by using two networks, say, $N(L_p)$ and $N(L_q)$, where $L_p * L_q = L_v$. The $3v$ primary input variables of $N(L_v)$ will simply be $x_1, x_2, \cdots, x_{3v}$. Similarly, the $3p$ ($3q$) primary input variables of $N(L_p)$ ($N(L_q)$) will be $x_1, x_2, \cdots, x_{3p}$ (correspondingly, $x_1, x_2, \cdots, x_{3q}$). However, when a network, say, $N(L_q)$ is used as a subnetwork, as in the construction of $N(L_v)$, then the $3q$ primary input variables of the subnetwork $N(L_q)$ will be some $3q$ consecutive x_j's, such as, $x_{3p+1}, x_{3p+2}, \cdots, x_{3p+3q}$.

The Procedure can now be formally stated as follows.

Network Construction Procedure:

Step 1: If $v = 1$, then $N(L_v)$ is as shown in Fig. 3(a). Stop.

Step 2: If $v \geq 2$, let $v = p + q$ such that $L_v = L_p * L_q$. Furthermore, assume that $N(L_p)$ and $N(L_q)$ are known from previous use of this procedure.

Step 3: Let the primary input variables of $N(L_p)$ be $x_1, x_2, \cdots, x_{3p}$, and that of $N(L_q)$ be $x_{3p+1}, x_{3p+2}, \cdots, x_{3p+3q}$.

Step 4: If $L_v = L_p + L_q (L_p L_q)$, then form $N(L_v)$ by ORing (ANDing) $N(L_p)$ and $N(L_q)$ as shown in Fig. 3(b) [see Fig. 3(c)]. Stop.

In the following we will assume that an input line which is connected to the primary input variable x_j, $1 \leq j \leq 3v$, is labeled with integer j. Note now that each $N(L_v)$ consists of exactly v subnetworks of the type shown in Fig. 3(a). Moreover, these v subnetworks are logically interconnected in $N(L_v)$ in the same manner as the v "L" symbols in L_v. Thus, it implies that if K is a set of v lines in $N(L_v)$ such that there is exactly one line in K from each of these v subnetworks, the L-expression associated with the set K would be L_v. We will thus be concerned in the following with the specific set $K = \{2, 5, \cdots, 3v - 1\}$ of v lines in $N(L_v)$.

We now present a test set generation procedure which will generate a SFDTS, T_s, for $N(L_v)$ such that $\phi(K,T_s) = \theta(L_v)$. T_s will consist of $4v$ input vectors, namely, $T_s = \{X_1^{01}, X_2^{01}, X_2^{10}, X_3^{10}, X_4^{01}, X_5^{01}, X_5^{10}, X_6^{10}, \cdots, X_{3k-2}^{01}, X_{3k-1}^{01}, X_{3k-1}^{10}, X_{3k}^{10}, \cdots, X_{3v-2}^{01}, X_{3v-1}^{01}, X_{3v-1}^{10}, X_{3v}^{10}\}$ where each $X_i^{01}(X_i^{10})$ is an input vector which $(0 \rightarrow 1)$-tests ($(1 \rightarrow 0)$-tests) the single line i-s-a-1 (line i-s-a-0) in $N(L_v)$. It will be assumed that the sets $P_{\min}^1(L_v)$ and $P_{\min}^0(L_v)$ from which $\theta(L_v)$ was calculated are given.

Test Set Generation Procedure:

Do Steps 1, 2, and 3 for $t = 1, 2, \cdots, v$.

Step 1: Let $c^t \epsilon P_{\min}^{\beta}(L_v)$, $\beta \epsilon \{0, 1\}$, be the cube which represents the set $D_t^{\beta}(L_t)$ in $P_{\min}^{\beta}(L_t)$. Moreover, assume that $c^t = (c_1^t c_2^t \cdots c_t^t)$.

Step 2: For a given β, do the following for $i = 1, 2, \cdots, v$. The resulting output will be X_{3i-1}^{01} if $\beta = 1$, and X_{3i-1}^{10} if $\beta = 0$.

a) If $c_i^t = \beta$, $t \neq i$, then let $x_{3i-2} = 1$, $x_{3i-1} = \beta$ and $x_{3i} = 0$;

b) If $c_i^t = \beta$, $t = i$, then let $x_{3i-2} = 1$, $x_{3i-1} = \bar{\beta}$ and $x_{3i} = 0$;

c) If $c_i^t = x$, then let $x_{3i-2} = x_{3i-1} = x_{3i} = \bar{\beta}$.

Step 3: For a given β, do the following for $i = 1, 2, \cdots, v$. The resulting output will be X_{3i-2}^{10} if $\beta = 1$, and X_{3i}^{10} if $\beta = 0$.

a) Same as Step 2a);

b) If $c_i^t = \beta$, $t = i$, then let $x_{3i-2} = 0$, $x_{3i-1} = \beta$, and $x_{3i} = \bar{\beta}$;

c) Same as Step 2c).

Stop.

The test set, T_s, obtained by the above procedure is an SFDTS for $N(L_v)$, and is such that $\phi(K,T_s) = \theta(L_v)$ [17]. The following example illustrates the above developments.

Example: Let $L_v = (L + L)(L + L)$. The network $N(L_v)$ obtained by using the Network Construction Procedure is shown in Fig. 4. Let $K = \{2, 5, 8, 11\}$ be the set of 4 lines of interest. Recall from the Example given in Section IV that for $L_v = (L + L)(L + L)$, we found that $\theta(L_v) = 14$, and

$$P_{\min}^1(L_v) = (1x1x, x1x1)$$

and

$$P_{\min}^0(L_v) = (00xx, xx00).$$

Completely generating T_s with the Test Set Generation Procedure will result in the specification of $4v = 16$ input vectors. We will, however, illustrate the use of the Test Set Generation Procedure only for the generation of one input vector. The complete T_s can be generated similarly and is listed in [17].

Thus, consider the single fault, say, $\alpha_8 = 0$ (that is, the fault line 8-s-a-0). Then to determine X_8^{10} we use Step 2 of the procedure for $t = 3$ and $\beta = 0$. Doing this, we note that the cube c^t, $t = 3$, which represent $D_t^0(L_v)$ in $P_{\min}^0(L_v)$ is $(xx00) = (c_1^3 c_2^3 c_3^3 c_4^3)$. Therefore, since $c_i^t = x$, for $i = 1$, we have from Step 2c) that

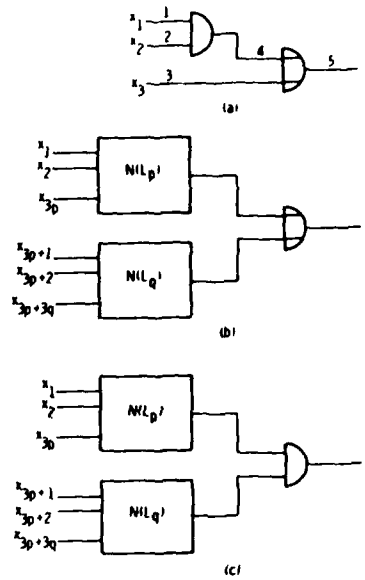


Fig. 3. Networks for the network construction procedure.

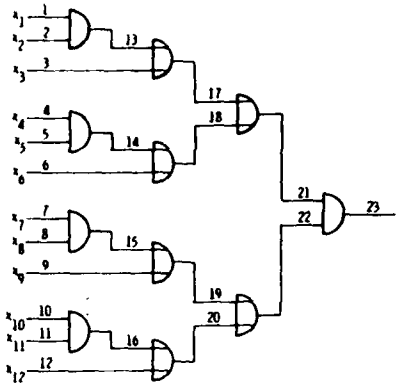


Fig. 4. Network for the example in Section V-B.

$$x_1 = 1, x_2 = 1, \text{and } x_3 = 1.$$

Similarly, since $c_i' = x$ for $i = 2$, we have

$$x_4 = 1, x_5 = 1, \text{and } x_6 = 1.$$

For $i = 3$, $c_i' = 0$ and $i = t$; thus, by Step 2(b) we have

$$x_7 = 1, x_8 = 1, \text{and } x_9 = 0.$$

Finally, for $i = 4$, we get from Step 2(a) that

$$x_{10} = 1, x_{11} = 0, \text{and } x_{12} = 0.$$

That is,

$$X_8^{10} = (1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0).$$

The procedure can similarly be applied to generate the remaining 15 input vectors.

It is important to point out here that although the networks constructed in this subsection are fan-out-free networks, it is also possible to construct networks which have primary input fan out and for which $\theta(L_v) = \phi(K)$. However, because the Network Construction Procedure and the Test Set Generation Procedure for such networks are much more complex, we have for simplicity considered fan-out-free networks to justify Property P2.

In conclusion, it has been shown in this section that given any set of K of v lines in any internal fan-out-free network N, the greatest lower bound on the minimum coverage $\phi(K)$ is simply the integer $\theta(L_v)$, where L_v is the L-expression associated with K and $\theta(L_v)$ is obtained by using the Coverage Algorithm.

VI. Conclusion

This paper develops a new theoretical approach to determining, quantitative bounds on the multiple fault coverage capability of single fault detection test sets in combinational internal fan-out-free networks. The importance of such bounds is underscored by the development of VLSI technology where it is becoming increasingly evident that explicit multiple fault coverage considerations are prohibitive. The main feature of our approach is the generic representation of faults in terms of L-expressions. Using this representation, it is possible to determine the greatest lower bound on the number of multiple faults covered by any SFDTS on any set of lines in any internal fan-out-free network by means of a reference into a precalculated Coverage Table. The generality of our results can be seen by noting that various previously known results of this type simply correspond to specific entries in the Coverage Table. Moreover, an averaging over all the entries of the Coverage Table yields general, quantitative observations, such as those stated in Theorems 1 and 2.

Although the discussions in this paper were limited to irredundant networks, the concept of L-expressions is equally applicable to redundant networks. Some preliminary work along these lines has been reported elsewhere [30]. Furthermore, there are various other interesting problems to which the application of the L-expression concept would be appropriate:

1) Since it can clearly be observed from the Coverage Table that there are certain inherently "bad" L-expressions in the sense that such L-expressions indicate network structures which can have multiple faults that are poorly covered by SFDTS's, it would be useful to develop design principles which minimize the occurrence of such L-expressions in combinational networks.

2) The averaging scheme used in this paper is based on a heuristic assumption of the "equinumerous occurrence" of L-expression. It is clear, however, that more network-specific averaging schemes would lead to tighter bounds.

3) The extension of these results to general reconvergent fanout network of Class *C3* would be significant. A possible approach to such an extension might be developed by introducing the concept of a labeled L-expression, in which all L's with the same label would correspond to a stem line and its fan-out branches. Similar to the approach taken in this paper, this labeled L-expression approach also would have the advantage of utilizing the well developed theory of labeled two-terminal series-parallel networks [20] to enumerate and specify various sets of labeled L-expressions;

4) The generic representation concept would be particularly interesting to explore under the constraints of multiple fault testing problems in highly structured devices such as PLA's and ROM's [29].

References

[1] J. P. Hayes, "The fanout structure of switching functions," *J. Ass. Comput. Mach.*, vol. 22, pp. 551-571, October 1975.

[2] ——, "Enumeration of fanout-free Boolean functions," *J. Ass. Comput. Mach.*, vol. 23, pp. 700-709, Oct. 1976.

[3] V. K. Agarwal, "Fanout-free Boolean functions and L-expressions," submitted to *J. Ass. Comput. Mach.*, for review. (Also, appeared in the *Proceedings of the 1978 Conference on Information Sciences and Systems*, The Johns Hopkins University, Baltimore, MD, 1978, pp. 227-233).

[4] G. Markowsky, "A straightforward technique for producing minimal multiple fault test sets for fanout-free circuits," IBM Res. Rep. RC 6222, 9/29/76, Yorktown Heights, NY.

[5] G. Markowsky and C. W. Cha, "No single fault test set is smaller than any minimal multiple fault test set for a fanout-free combinational circuit," IBM Rep. RC6483, 4/13/77, Yorktown Heights, NY.

[6] S. C. Seth and K. L. Kodandapani, "Diagnosis of faults in linear tree networks," *IEEE Trans. Comput.*, vol. C-26, pp. 29-33, Jan. 1977.

[7] C. D. Latino and J. G. Bredeson, "Simplified multiple stuck-at-fault test generation techniques," in *Proc. 13th Annu. Allerton Conf.*, 1975, pp. 682-691.

[8] J. P. Hayes, "A NAND model for fault diagnosis in combinational logic networks," *IEEE Trans. Comput.*, vol. C-20, pp. 1496-1506, Dec. 1971.
[9] V. K. Agarwal and G. M. Masson, "A functional form approach to test set coverage in tree networks," *IEEE Trans. Comput.*, vol. C-28, pp. 50-52, Jan. 1979.
[10] V. K. Agarwal and G. M. Masson, "Recursive coverage projection of test sets," *IEEE Trans. Comput.*, vol. C-28, Nov. 1979.
[11] J. W. Gault, J. P. Robinson, and S. M. Reddy, "Multiple fault detection in combinational networks," *IEEE Trans. Comput.*, vol. C-21, pp. 31-36, Jan. 1972.
[12] R. J. Diephius, "Fault analysis for combinational logic networks," Ph.D. dissertion, Dept. of Elec. Eng., Mass. Inst. Tech., Cambridge, MA, Sept. 1969.
[13] D. R. Schertz and G. Metze, "On the design of multiple fault diagnosable networks," *IEEE Trans. Comput.*, vol. C-20, pp. 1361-1364, Nov. 1971.
[14] D. B. Armstrong, "On finding a nearly minimal set of fault detection tests for combinational logic sets," *IEEE Trans. Comput.*, vol. C-15, pp. 66-73, Feb. 1966.
[15] C. T. Ku and G. M. Masson, "The Boolean difference and multiple fault analysis," *IEEE Trans. Comput.*, vol. C-24, pp. 62-71, Jan. 1975.
[16] L. H. Goldstein, "A probablistic analysis of multiple faults in LSI circuits," IEEE Computer Soc. Rep. R-77-304, IEEE Computer Society, Long Beach, CA.
[17] V. K. Agarwal, "Generic characterizations of multiple faults for table-look-up coverage bounding in tree networks," Ph.D. dissertion, Dept. Elec. Eng., Johns Hopkins Univ., Baltimore, MD, Sept. 1977.
[18] P. A. MacMohan, "The combination of resistances," *Electrician*, Apr. 8, 1892.
[19] J. Riordan and C. E. Shannon, "The number of two-terminal series-parallel networks," *J. Math. Phys.*, vol. XXI, pp. 83-93, 1942.
[20] J. Riordan, *An Introduction to Combinational Analysis*. New York: Wiley, 1958.
[21] D. L. Dietmeyer, *Logic Design of Digital Systems*. Boston, MA: Allyn and Bacon, 1971.
[22] M. A. Breuer, "Logic synthesis," in *Design Automation of Digital Systems*, Vol. 1, M. A. Breuer, Ed. Englewood Cliffs, NJ: Prentice-Hall, 1972.
[23] E. B. Eichelberger and T. W. Williams, "A Logic Design Structure for LSI Testability," in *Proc. 14th Design Automation Conf.*, New Orleans, LA, IEEE Catalog Number 77 CH1216-1C, pp. 462-468, June 1977.
[24] E. J. McCluskey and F. W. Clegg, "Fault equivalence in combinational logic networks," *IEEE Trans. Comput.*, vol. C-20, pp. 1286-1293, Nov. 1971.
[25] D. R. Schertz and G. Metze, "A new representation for faults in combinational digital circuits," *IEEE Trans. Comput.*, vol. C-21, pp. 858-866, Aug. 1972.
[26] C. W. Cha, "Multiple fault diagnosis in combinational networks," Ph.D. dissertation, R-650, Coordinated Science Lab., Univ. Illinois, June 1974.
[27] D. C. Bossen and S. J. Hong, "Cause effect analysis for multiple fault detection in combinational networks," *IEEE Trans. Comput.*, vol. C-20, pp. 1252-1257, Nov. 1971.
[28] I. Berger and Z. Kohavi, "Fault detection in fanout-free combinational networks," *IEEE Trans. Comput.*, vol. C-22, pp. 908-914, Oct. 1973.
[29] V. K. Agarwal, "Multiple fault detection in programmable logic arrays," in *Proc. 1979 Int. Symp. Fault-Tolerant Computing*, Madison, WI, pp. 227-234, June 1979.
[30] V. K. Agarwal and G. M. Masson, "Redundancy results in combinational networks by means of *L*-expressions," in *Proc. Fifteenth Annual Allerton Conf. Communication Control and Computing*, Urbana-Champaign, IL, pp. 732-742, Sept. 1977.

Vinod K. Agarwal (S'75-M'77-SM'79) was born in Mathura, India, on April 30, 1952. He received the B.E. (Hons.) degree in electronics engineering from the Birla Institute of Technology and Science, Pilani, India, in 1973, the M.S. degree in electrical engineering from the University of Pittsburgh, Pittsburgh, PA, in 1974, and the Ph.D. degree in electrical engineering from The Johns Hopkins University, Baltimore, MD, in 1977.

From 1977 to 1978, he was an Assistant Professor in the Department of Electrical and Computer Engineering, Wayne State University, Detroit, MI. Presently he is an Assistant Professor in the Department of Electrical Engineering, McGill University, Montreal, P.Q., Canada. His teaching and research interests are in switching theory, computer structure and organization, operating systems, and fault-tolerant computing.

Gerald M. Masson (S'67-M'71-SM'78) was born in Chicago, IL on May 5, 1943. He received the B.S.E.E. degree in 1966 from the Illinois Institute of Technology, Chicago, and the M.S. and Ph.D. degrees in electrical engineering from Northwestern University, Evanston, IL, in 1968 and 1971, respectively.

In 1971 he joined the Department of Electrical Engineering, University of Pittsburgh, Pittsburgh, PA. In 1975 he joined the Department of Electrical Engineering, The Johns Hopkins University, Baltimore, MD, where he is currently an Associate Professor. His research interests include fault tolerant computing, interconnection networks, and computer system design.

This overview presents several different types of circuit switching networks: concentrators, connectors, expanders, partitioners, SIMD interconnectors, and sorters.

A Sampler of Circuit Switching Networks

Gerald M. Masson
The Johns Hopkins University

George C. Gingher
Bethlehem Steel Corporation

Shinji Nakamura
The Johns Hopkins University

Circuit switching networks are *systems* which provide a set of interconnecting circuits from a set of inputs to a set of outputs by opening and closing switches, or crosspoints. As a discipline, circuit switching networks deal fundamentally with the design and analysis of crosspoint patterns. At first thought, the idea of designing a system for simply interconnecting terminals might seem too basic to constitute a research area. However, the vast majority of the area of circuit switching networks lies far beneath the surface.

This paper presents some of the important results of recent research into circuit switching networks and their complexity. Several different types of circuit switching networks will be discussed, together with the theoretical complexity and the best known explicit constructions needed to implement them. As with any sampler, the network designs we display will hardly exhaust the available possibilities. Nor do we mean to imply that those omitted are held in lower regard than those we discuss. Our intent is only to show that the area of circuit switching networks is embroidered with many fascinating designs which are now, more than ever before, applicable to computer system architectures.

Much of the early research on circuit switching networks was motivated by the needs of the communications industry. Indeed, a great deal of the work in this area still is pursued because of that application. However, with the advent of LSI/VLSI technologies, a circuit switching network now often represents a principal subsystem in a large class of memory/processor/peripheral computer complexes. In fact, in many such systems the associated circuit switching network significantly, and even dominantly, affects the overall cost and performance of the system.

As generic examples of their uses in these applications, three schematics of computer systems containing circuit switching networks are shown in Figure 1. In Figure 1(a), the circuit switching network is required to provide paths between a large set of input devices to some smaller set of output ports. These output ports then provide access for those input devices to some other devices or functions. It is usually not cost-efficient or necessary for general operation to have one such device or function for each input device. Hence, subsets of the input devices are selected by the circuit switching network and, in effect, concentrated to the output side of the network as required by system operation. In some cases, the output functions can all be identical so that the order in which the input devices are interconnected to the output ports becomes unimportant. Such an interconnection requirement, for example, can be found in the area of image processing where the input devices are memory cells and the output functions are correlations, or where the input devices are smart sensors and the output functions are signal processors.

In Figure 1(b), the circuit switching network is required to provide paths between specified devices at the input side to specified devices or sets of devices at the output side. Some or all of the devices at the input and output sides might actually be identical, but the general operation can involve them in such a way that they must be treated as though they were distinguishable. Hence, the input devices are connected in a one-to-one manner or expanded in a one-to-many manner by the network to the output devices as required by system operation. Such an interconnection requirement can be found in numerous parallel processing environments. For example, if the input devices are processors and the output devices are

0018-9162/79/0600-0032$00.75

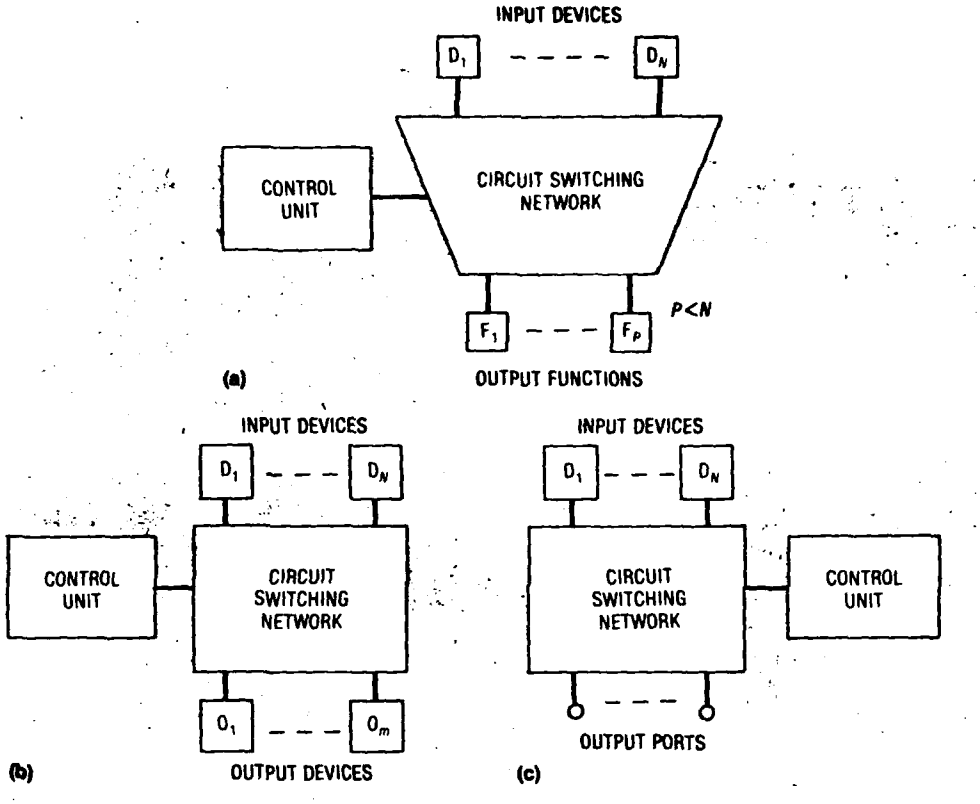


Figure 1. Generic examples of computer systems containing circuit switching networks: (a) concentration, (b) connection/expansion, and (c) partitioning.

memory units, then we have what is often referred to as a data alignment/data access interconnection requirement.

Finally, in Figure 1(c), the circuit switching network must partition the set of input devices into disjoint subsets so that the subsets can function as independent subsystems. Here, many different partitioning requirements are possible depending on whether or not the devices are distinguishable and whether or not the specific output ports to which they are interconnected is important.

Terminology

A circuit switching network can be described as a set of contact switches, generically called *crosspoints*, joined together by links in order to interconnect network input terminals (inputs) to network output terminals (outputs). Circuit switching networks of interest must be capable of doing this for a very large number of interconnection requirements. An example of such a network is explicitly shown in Figure 2(a) for N-inputs and M-outputs. This network is known as an *$(N \times M)$-complete crossbar network*, or, more simply, as an *$(N \times M)$*-network. Clearly, the number of crosspoints is NM, since there is one crosspoint between each of the inputs and each of the outputs. As we will see, many well-known circuit switching network constructions consist of interconnected complete crossbar networks.[1-4]

However, not all crossbar switches of interest are complete. A sparse crossbar switch has some inputs and outputs with no crosspoint between them and, consequently, between which no direct interconnec-

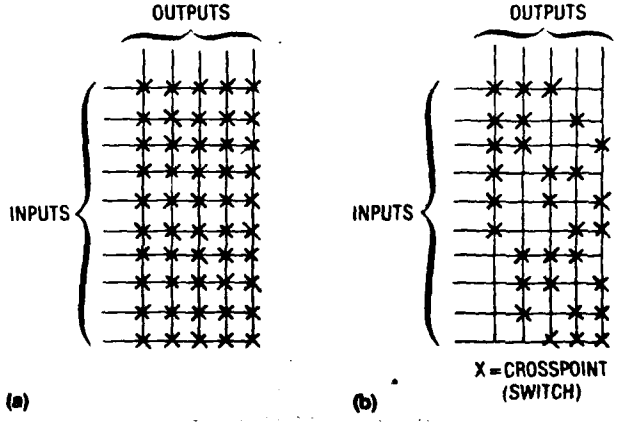


Figure 2. (a) complete (N x M)-crossbar switch; (b) $\binom{5}{3}$-binomial network.

tion is possible. An example of this type of construction is the binomial network[5] shown in Figure 2(b).

Besides the illustrations of Figures 2(a) and 2(b), which explicitly show inputs, outputs, and crosspoints, another means of describing crossbar networks is to model them as bipartite graphs. In addition, the adjacency matrix of a bipartite graph can be used to represent crossbar networks where the ones represent crosspoint placements. This is illustrated in Figure 3. In Figure 3(a), a (4×3)-network is shown explicitly; Figure 3(b) is a node-edge representation of the network, with the 12 edges representing the 12 crosspoints; and in Figure 3(c) we have the corresponding graph adjacency matrix. If it is necessary to indicate the direction of signal flow in the graph, the edges of the graph representing the network can be directed from inputs to outputs. Henceforth, we will illustrate our circuit switching networks with whichever representation is most convenient to the immediate discussion.

Circuit switching networks are usually classified according to the types of interconnections which they must provide. In the following sections we will discuss some aspects of the circuit switching networks known as concentrators, connectors, expanders, partitioners, SIMD interconnectors, and sorters. Definitions of the input to output interconnection requirements which each of these networks satisfy are treated in their respective sections. For each such type of circuit switching network, however, two important aspects relating to their interconnecting capabilities should be pointed out here. Note that the circuit switching network of Figure 3 can interconnect all possible combinations of inputs to outputs; and it can provide any new additional interconnection between an unused (or "idle") input to a not necessarily idle output regardless of the other interconnections which are currently in progress. A network with this property is said to be *nonblocking*. In other words, a nonblocking network can satisfy any new, valid interconnection request in the sense of providing a path for that request without disturbing any of the other existing interconnections in the network. This idea can be made somewhat more formal by noting that a specification of the input-to-output interconnection requirements at any time in a network is often referred to as an *interconnection assignment*, and it is sometimes described as a mapping of the inputs onto the outputs. A total assignment includes all the inputs/outputs. A detailing of the opened and closed crosspoints at any time in a network is referred to as the state of the network. Then, a nonblocking network in any state satisfying some valid interconnection assignment can satisfy any new, valid interconnection assignment containing the previous assignment by being placed in a new state containing the old state. It does this by closing crosspoints to provide the additional interconnecting paths without any disturbance to existing paths.

As opposed to the above, suppose that the network interconnection requirement is to provide disjoint interconnections from any specified set of inputs to some specified set of the outputs. Suppose further that, for some assignment where interconnections are in progress, an additional request can sometimes only be realized by rearranging some of the existing interconnecting paths. A network which provides interconnections in this manner is said to be a *rearrangeable network*. Hence, placing a rearrangeable network in a new state to satisfy an interconnection assignment containing the presently satisfied interconnection requirement sometimes requires the disturbance of existing paths. In such networks, the upper bound on the maximum number of rearrangements necessary to provide for any additional interconnection is usually significant.

Two of the main circuit switching network design techniques are *decomposition* and *staging*. The former consists of decomposing the initial interconnection requirement into a set of subrequirements, each of which is similar in nature to the initial requirement but usually smaller in size and, therefore, less complex to realize. This decomposition is usually continued on the subrequirements to produce even finer subrequirements. This is then recursively continued until it is feasible to satisfy the finest subrequirement with a brute-force or obvious design. Proceeding backwards relative to the decomposition technique towards the initial interconnection requirement effectively specifies a structure which satisfies the original requirement. The overall structure is then referred to as a *multi-stage structure*. Each stage will often consist of a set of subnetworks linked to the subnetworks of other stages. For example, an output terminal of a subnetwork of one stage might be linked to an input terminal of a subnetwork of another stage.

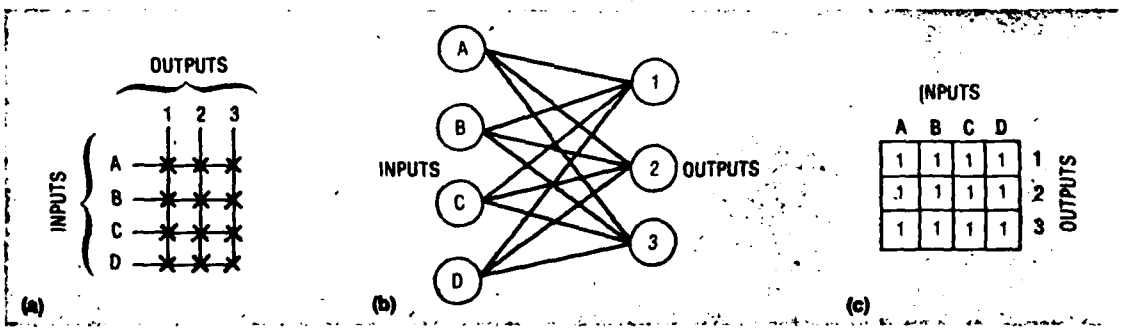


Figure 3. Representations of a complete crossbar switch: (a) crossbar, (b) bipartite graph, G, (c) adjacency matrix of G.

In the following, in illustrations of multi-stage networks showing explicit crosspoint designs, the links will be shown as solid lines connecting the terminals (input and output) of one stage with the terminals of other stages. Such links can simply be viewed as hardwired interconnectors. In the graph models of interconnection networks, the links will either be shown as dotted lines (so as not to confuse them with edges which correspond to crosspoints), or, when appropriate, the output nodes of one stage will be superimposed on the input nodes of the stage to which they would be linked in an explicit construction. When discussing graph models, we will often refer to the input/output nodes of a model as simply the inputs/outputs of the network, and the input/output nodes of a stage as simply the inputs/outputs of the stage.

Implicit in the concept of a multi-stage design of a circuit switching network is a tradeoff which is at the center of much of the research in the area. Namely, we will be motivated to consider multi-stage designs for various types of interconnection requirements because such designs will often be shown to employ far fewer crosspoints than straightforward, one-stage alternatives. However, the overall reduction in crosspoints will be paid for by the added complexity of realizing assignments in the network in the sense of computing states to satisfy requests, and in the delays of the network in the sense that a signal flowing from the network input to the network output will be required to pass through more than one crosspoint. This size versus control conflict will be seen repeatedly in the following sections.

Accordingly, the role of complexity theory in providing bounds on the maximum and minimum number of crosspoints required in order to realize a given interconnection requirement is significant in the area of circuit switching networks. Lower bounds provide a theoretical minimum, below which it is futile to attempt to construct practical networks; and upper bounds provide a convenient benchmark with which to measure proposed designs. In the following, for each type of circuit switching network, we will give the known theoretical and constructional bounds. For those bounds, all logarithms will be to the base 2 unless otherwise specified. It is interesting to point out here that along these lines we will see cases in the following sections where a decomposition of a class of assignments can be described, and the existence of a subnetwork which satisfies the assignments with a given bound on its number of crosspoints can be proven, but where no explicit practical construction technique for designing these networks is known.

Concentrators

A concentration circuit switching network or, more simply, a concentrator interconnects specific idle inputs to arbitrary idle outputs by providing disjoint circuits from the inputs to the outputs[5-7] Clearly, only concentrators where N, the number of inputs, is greater than M, the number of outputs, are of interest. We will let the capacity of a concentrator be denoted by R, where R is the maximum number of interconnections that can be made simultaneously through the network. In other words, an N input to M output concentrator of capacity R is a circuit switching network capable of interconnecting any of the $\binom{N}{K}$ choices of inputs, $K \leq R$, to some K of the outputs. The important point is that the inputs can be a priori specified but the outputs to which they will be connected cannot be so specified. Hence, a concentrator can be functionaliy described by a triplet of integers $(N, M, R,)$, $N > M \geq R$, where

N = number of inputs,
M = number of outputs,
R = capacity.

Figure 3 is a (4×3)-network which is functionally a (4,3,3)-nonblocking concentrator. Indeed, it is clear that an $(N \times M)$-network is an $(N,M,M,)$-nonblocking concentrator. However, concentrators with fewer than NM crosspoints are known. The second stage of Figure 5 is a (6,4,4)-rearrangeable concentrator in which the pattern of crosspoints connecting the outputs to the inputs consists of all the possible $\binom{4}{2}$ choices of two crosspoints between the inputs and the four outputs. This configuration, called a *binomial concentrator*, can be shown to have a capacity of 4.[5]

Concentrators were first defined by Pinsker,[6] who proved that there exist (N,M,M)-rearrangeable concentrators for all $N > M$ with at most $29N$ crosspoints. One particularly interesting aspect of Pinsker's network, as shown in Figure 4, is his recursive use of a network structure in order to obtain this bound. As discussed earlier, it is common to exploit a structural concept recursively, starting with the complete problem and using some insightful observation to form subproblems which can be further decomposed. The key, of course, is to determine the structural concept which provides some reduction in, for this case, the number of crosspoints (or edges in the graph model) relative to a benchmark and, perhaps, even agrees asymptotically with the information theoretical optimum.

Thus, an upper bound on the necessary asymptotic growth of the number of crosspoints in an (N,M,M)-rearrangeable concentrator as the number of inputs, N, grows is on the order of N crosspoints, that is, $O(N)$ crosspoints. Relative to the information theoretical bound, this is the best order of growth possible. However, it must be pointed out here that Pinsker's results are non-constructive in that they are not generally based on explicit constructions. Rather, they are based on a proof of only the existence of an (N,M,R)-rearrangeable concentrator for all $R < M$. It is significant that there does not exist a method at present (other than an exhaustive consideration of all possibilities) which leads to their explicit construction for all N.

There have, however, been some other recent contributions regarding constructions. Margulis,[8] on the basis of a complex argument involving the theory of group representations, has given a construction

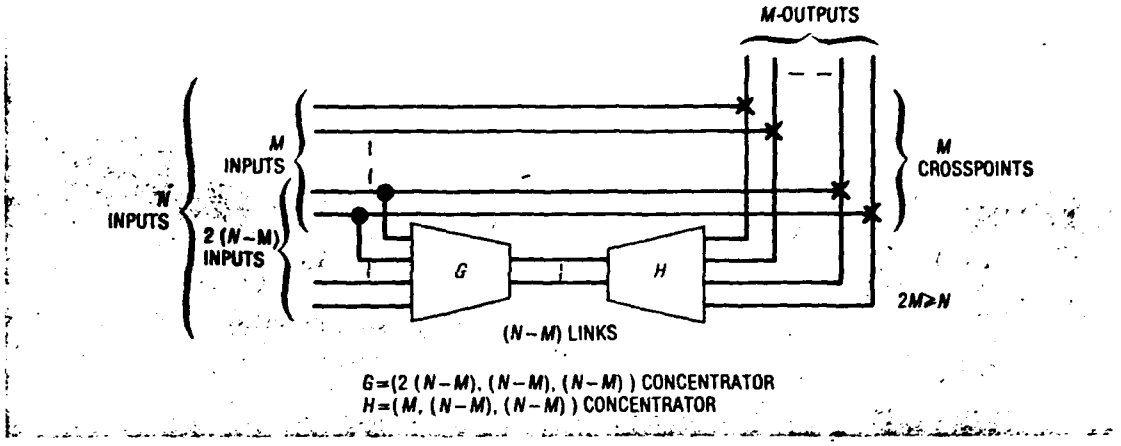


Figure 4. Pinsker's concentration network.

technique which results in a concentrator of predetermined capacity with K stages, wherein each stage requires at most $5N$ crosspoints. However, we are not aware of any results that firmly establish a bound for K. This unfortunately jeopardizes the applicability of Margulis' results.

A known constructive result is Masson's binomial concentrator[5] (see Figure 5), which shows that the $\binom{6}{2}$-network is a (15,6,4)-rearrangeable concentrator and that the $\binom{4}{2}$-network is a (6,4,4)-rearrangeable concentrator. Hence the cascade of Figure 5 is a (15,4,4)-rearrangeable concentrator with only 42 crosspoints. This can be favorably compared with a benchmark of a complete bipartite graph with 15 inputs and four outputs representing a concentrator which has 60 crosspoints. A practical aspect of the use of the binomial construction is that an upper bound of one plus the number of stages can be shown to exist for the number of existing paths which must be rearranged to satisfy a new request in its operation.

In general, the graph model of an $\binom{X}{Y}$-binomial circuit switching network or, more simply, an $\binom{X}{Y}$-network, has $\binom{X}{Y}$-inputs, X-outputs, Y-edges/input, $Y\binom{X}{Y}/X$ edges/output, and a capacity of $Y+2$.

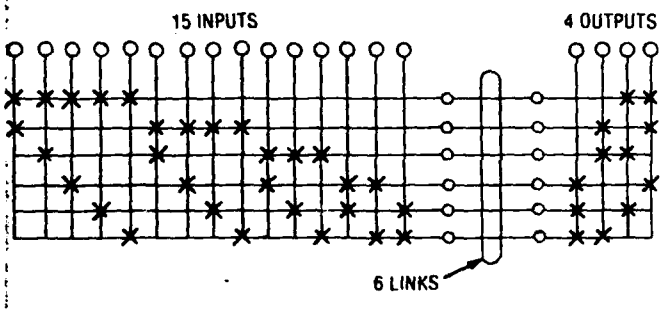


Figure 5. Masson's binomial concentrator.

Thus it can functionally be described as an $(\binom{X}{Y}, X, Y+2)$-rearrangeable concentrator.

Another construction of a concentrator is an interesting generalization of the basic binomial network. Recall that the adjacency matrix of the graph representing a network is composed of zeros and ones, where each of the non-zero entries denote the position of edges between the vertices of the graph corresponding to the inputs and outputs of the network. Suppose now that instead of each vertex of this graph representing only a single input or output terminal, it represented a set of input or output terminals, and in the same sense, suppose that each edge in this graph model represented a binomial network providing interconnecting paths between those input terminals and output terminals. The capacity of the resulting composite network would clearly be a function of the base network and the particular replacement networks used for the replacement of each nonzero in the graph adjacency matrix. For example, if the graph of a $\binom{4}{2}$-network is used not only as the base graph, but also as the replacement, the result is a (36,16,11)-rearrangeable concentrator with only 144 edges.[9] Note that a 36-input, 16-output network with 11 crosspoints between each input and the outputs would require 396 crosspoints.

Such a network corresponds to a super binomial network. More explicitly, it can be described as a $\binom{4}{2}^2$-network. In general, any $\binom{Y+2}{Y}^2$ graph, where $Y \geq 2$, has capacity of $R = (Y+1)(Y+2)-1$, and thus it represents a $(\binom{Y+2}{Y}^2, (Y+2)^2, (Y+1)(Y+2)-1)$-rearrangeable concentrator.[9] Such concentrators are attractive because they are sparse, and they are modeled as a single, incomplete bipartite graph. The more general class of $\binom{X}{Y}^n$ networks can be designed in an analogous manner.

Regarding nonblocking concentrators, Pippenger[7] has derived the (somewhat surprising) result that the number of crosspoints of such networks is $O(N \log N)$. As we will see, this is asymptotically the same as that required for nonblocking connectors, although the former intuitively seems as though it should be much

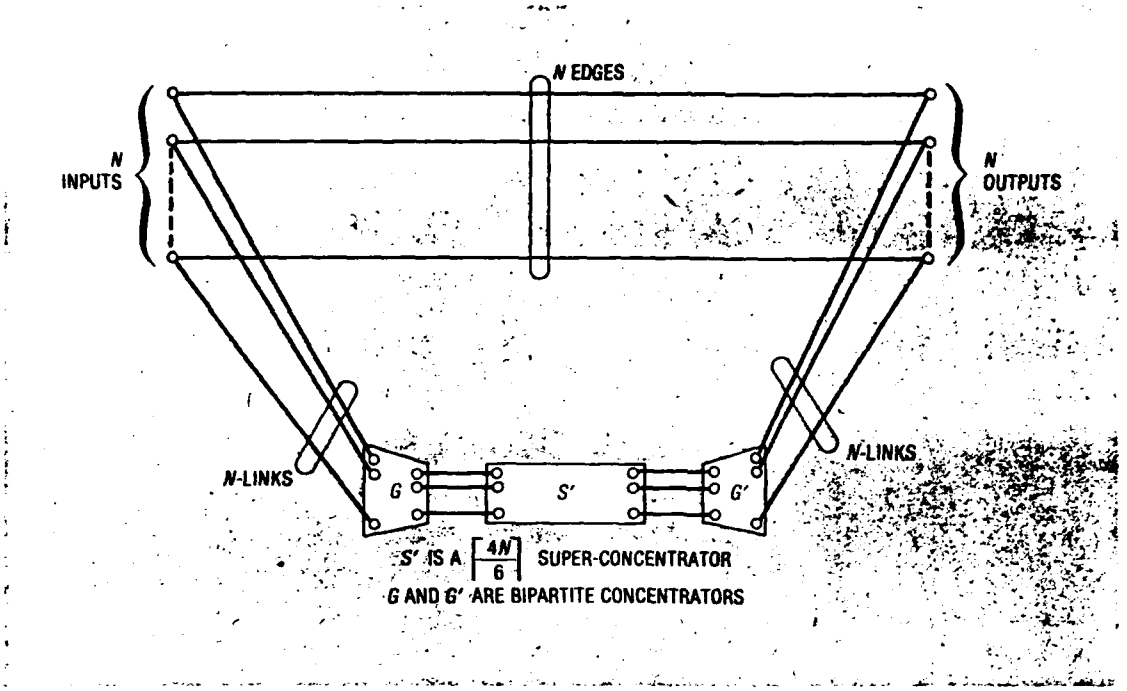


Figure 6. Pippenger's *N*-superconcentrator.

less difficult to implement than the latter. Little is known about minimal explicit constructions for nonblocking concentrators.

N-superconcentrators[10,11] are a type of rearrangeable circuit switching network with more powerful interconnecting capabilities than a concentrator. In the graph model of an *N*-superconcentrator with N inputs and N outputs for every set of $R \leq N$ inputs and every set of $R \leq N$ outputs, there exists a set of R vertex disjoint paths from the set of inputs to the set of outputs. Thus, in *N*-superconcentrators the sets of input and output terminals to be interconnected can both be specified, but the individual input-to-output interconnections within the sets cannot be specified. Clearly, superconcentrators are significantly more powerful than regular concentrators, yet Pippenger[11] has proven the existence of *N*-superconcentrators with at most $40N$ crosspoints. Again, however, no explicit construction which generally satisfies the bound is known. Pippenger's proof is based on the existence of (6m,4m,3m)-rearrangeable concentrators with only 36m crosspoints. These concentrators are used in the recursive construction illustrated in Figure 6.

It is worth pointing out here that Pippenger establishes his results on superconcentrators with a probabilistic approach. The principal idea behind such an approach is to show that the probability that a network exists is greater than zero, or conversely, the probability that it does not exist is less than one. Proving either conjecture implies that the network exists, although it does not explicitly lead to a construction. Besides Pippengers's paper,[11] a more thorough treatise of this type of probabilistic approach is provided by Erdos and Spencer.[12]

Another type of concentrator which is less powerful than an *N*-superconcentrator is an *N*-hyperconcentrator, which is a rearrangeable circuit switching network with N inputs and N outputs in which any specified set of $k \leq N$ inputs can be interconnected in some arbitrary order to the first k outputs. Clearly, an *N*-hyperconcentrator has less interconnecting capability than that of an *N*-superconcentrator. Thompson[13] has shown that a network called an (N,N)-infrageneralizer which is due to Ofman[14] can be reversed (in the sense that inputs of the (N,N)-infrageneralizer are considered as the outputs of the *N*-hyperconcentrator and the outputs of the (N,N)-infrageneralizer are considered as the inputs of the *N*-hyperconcentrator) to produce an *N*-hyperconcentrator. An 8-hyperconcentrator is illustrated in Figure 7. It is easily shown that such *N*-hyperconcentrators have $2N\log N$ crosspoints.[13] This particular construction of an *N*-hyperconcentrator has the enhanced capability of interconnecting the selected $k \leq N$ inputs to any $k \leq N$ consecutive outputs (allowing for wraparound) while maintaining the order of the inputs.

Connectors

In a connection circuit switching network, specific, idle inputs request interconnecting paths to specific, idle outputs. Because of the affinity of such networks

with telephony, much has been published on this subject. Moreover, if N is the number of inputs to a connector and M is the number of outputs, then when $N=M$, a connection assignment of all N inputs onto all N outputs can be represented by a bijective mapping of the inputs onto the outputs; in other words, the assignment is a permutation of the inputs. This observation permits the application of a broad range of mathematical theory to the design of connectors. Since results for other situations follow from the results for the case when $N = M$, we will discuss this case only in the following.

It is clear that since a complete crossbar network has one crosspoint between every input and output, such a network will operate as a nonblocking connector. Hence, if we refer to an N input, N output, connection network of capacity N as an (N,N,N)-(rearrangeable or nonblocking) connector, an upper bound on the number of crosspoints of such a connector is N^2. This then is a benchmark with which to compare our subsequent reported results.

To get an information theoretical lower bound on the order of growth of the number of crosspoints, we can use an argument given by Shannon.[15] Since there are effectively $N!$ assignments which must be realized by an (N,N,N)-connector, and since each crosspoint in such a network effectively has two states, opened or closed, 2 raised to the power the number of crosspoints must be greater than or equal to $N!$ From Stirling's well-known formula for the approximation of a factorial,[16] it follows that the asymptotic growth of the number of crosspoints is $O(N\log N)$. We will now compare some designs with this information theoretical lower bound on the number of crosspoints.

We will first consider rearrangeable connection networks. Benes' classic book[1] describes a construction which is based on Hall's theorem[17] (sometimes called "the marriage theorem"). For a three-stage network, the approach uses Hall's theorem to decompose the original assignment describing the mapping of the N inputs onto the N outputs into n subassignments. Repeating this decomposition on the subassignments leads to a multistage connector. The structure of such a network is explicitly shown in Figure 8 in a three-stage form. In this form, the structure consists of the following: (1) a first stage (also called an input stage) of r_1 $(n_1 \times m)$-networks, where the $n_1 r_1 = N$ terminals on the left (input) side of this stage are the inputs to the overall network; (2) a last stage (also called an output stage) of r_2 $(m \times n_2)$-networks where the $n_2 r_2 = N$ terminals on the right (output) side of this stage are the outputs to the overall network; (3) a middle stage of m $(r_1 \times r_2)$-networks where the terminals on the input side of this middle stage are linked to the terminals on the output side of the input stage so that each of the m $(r_1 \times r_2)$-networks in this middle stage has exactly one link to each of the r_1 $(n_1 \times m)$-networks in the input stage, and where the terminals on the output side of this middle stage are linked to the terminals on the input side of the output stage so that each of the m $(r_1 \times r_2)$-networks in this middle stage has exactly one link to each of the r_2 $(m \times n_2)$-networks in the output stage. It can be shown for this network[1] that if $m = \max(n_1, n_2)$, then the result is an (N,N,N)-connector. Note that if $n_1 = r_1 = n_2 = r_2 = N^{1/2}$, then the number of crosspoints in the network is $3N^{3/2} - 3N$. Moreover, it can be shown[18] that the maximum number of interconnecting paths through the network which must be rearranged to satisfy a new interconnection request from an idle input to an idle output is, in general, for such three-state structures $(\max(r_1, r_2) - 1)$.

Beizer,[19] Joel,[20] Waksman,[21] and Opferman and Tsao-Wu[22] give refinements of this approach with a decomposition which employs the repeated use of (2×2)-networks as the basic element of the network. The results are constructions in which the number of crosspoints has $O(N\log N)$ growth with N, which by our above discussion is the best order of growth possible. For example, in the (N,N,N)-Waksman connector, shown for the $N=8$ case in Figure 9, there are $4N\log N - 4N + 4$ crosspoints in general. Benes[1] has also shown that if the decomposition utilizes a (3×3)-network, then the number of crosspoints of $3.8N\log N$ crosspoints grows. It is worth noting that an (N,N,N)-Waksman network has fewer crosspoints than the $(N \times N)$-network for all $N \geq 16$.

It should be clear that for these networks the number of stages, and, therefore, the number of crosspoints involved in any input-output path, often referred to as the delay, will be $O(\log N)$. For example, the (N,N,N)-Waksman connector has a $2\log N - 1$ delay in this sense. This means, of course, that a realizing state to satisfy a request can be quite complex. Effectively, the use of a minimal crosspoint design implies that a tradeoff has been accepted resulting in fewer crosspoints at the cost of greater control complexity and delay. Regarding this latter point somewhat more specifically, given a request to be satisfied by one of the above $O(N\log N)$ connection networks, Waksman[21] and Tsao-Wu and Opferman[22]

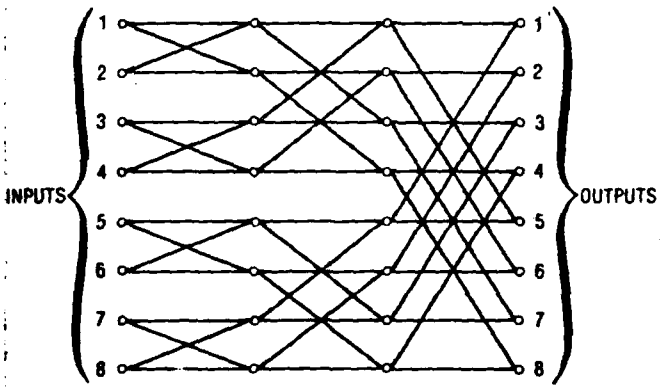


Figure 7. An 8-hyperconcentrator.

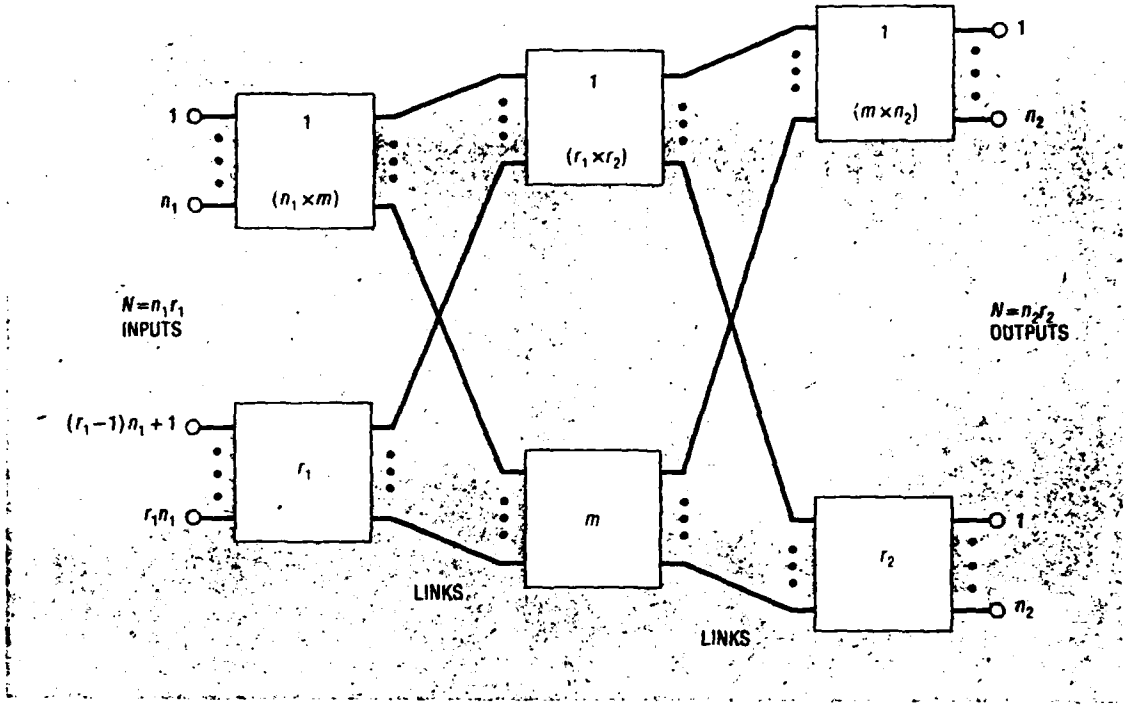


Figure 8. A three-state interconnection network structure.

give algorithms which require $O(N\log N)$ operations. Recently, Lenfant[23] has developed some control algorithms which take advantage of an apriori knowledge of "frequently used" requests to alleviate this control problem somewhat. However, for some real-time applications, set-up times might still be prohibitive. Indeed, delay and set-up time perhaps represent the most serious limitation of $O(N\log N)$ designs of (N,N,N)-rearrangeable connectors.

There are applications where not all mappings of the inputs to the outputs are of interest. Lawrie[24] has proposed what is referred to as an *omega network* for interconnections required in the access and alignment of data in an array processor. The omega network structure is effectively the left-hand half (including the middle stage) of the Waksman structure where the linking between the stages is usually in a perfect-shuffle[25] form. Clearly, not all permutations of the inputs to the outputs can be realized. However, Lawrie shows that the data elements of an $(n \times n)$ matrix can be stored using a skewing scheme in $2n$ independent memory units (each data element being placed in only one such unit) so that, in a conflict-free manner, rows, columns, diagonals, backward diagonals, and square blocks in row-major or column-major order can be accessed and changed by an omega network.

We next consider nonblocking connection networks. The first published work regarding the construction of such networks is due to Clos.[26] By means of a straightforward argument, he showed that for three stages of the structure of Figure 8, if $m \geq n_1+n_2-1$, the network has nonblocking capability. Note that if $n_1=r_1=n_2=r_2=N^{1/2}$ then $m=2N^{1/2}-1$, and the number of crosspoints in the resulting net-

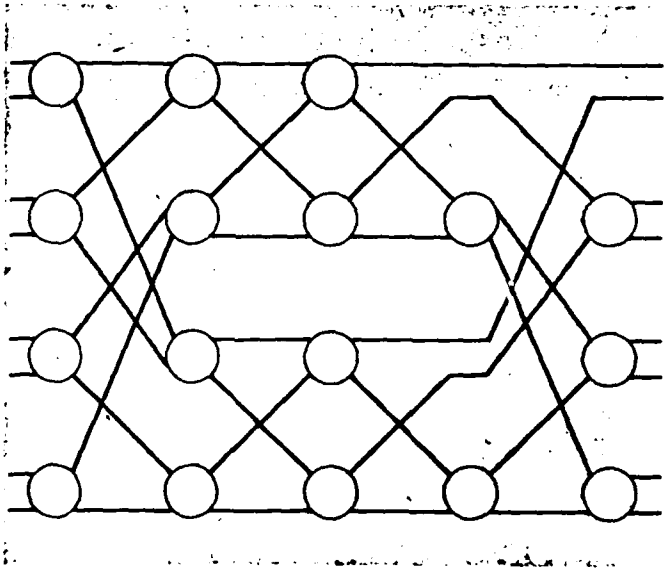


Figure 9. An (8,8,8)-Waksman connector.

work is $6N^{3/2} - 3N$. It is easily seen that the three-stage nonblocking Clos connector has fewer crosspoints than an $(N \times N)$-network for $N \geq 36$. Furthermore, Cantor[27] shows that with the Clos argument applied recursively, a structure with $O(N(\log N)^{2.269})$ results.

Bassalygo and Pinsker[28] have shown that, indeed, constructions for (N,N,N)-nonblocking connectors do exist for which the number of crosspoints is $O(N \log N)$. However, these results are non-constructive in the sense previously described, because their proof of the existence of such (N,N,N)-nonblocking connectors is based on a proof of the existence (but not the construction) of an N input, N output sparse network where each input has crosspoints between it and 12 of the N outputs and through which any choice of one-third of the inputs can be connected to more than two-thirds of the outputs. Such sparse networks could be linked together (if their explicit constructions were known) to yield an (N,N,N)-nonblocking connector with $O(N \log N)$ crosspoints. More precisely, Pippenger[29] has recently shown that such networks exist with $90N \log_3 N$ crosspoints.

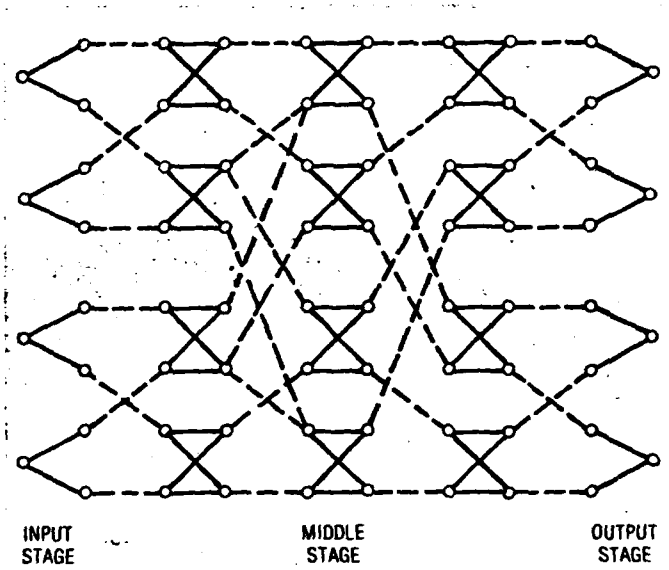


Figure 10. (4,4,4)-Cantor network.

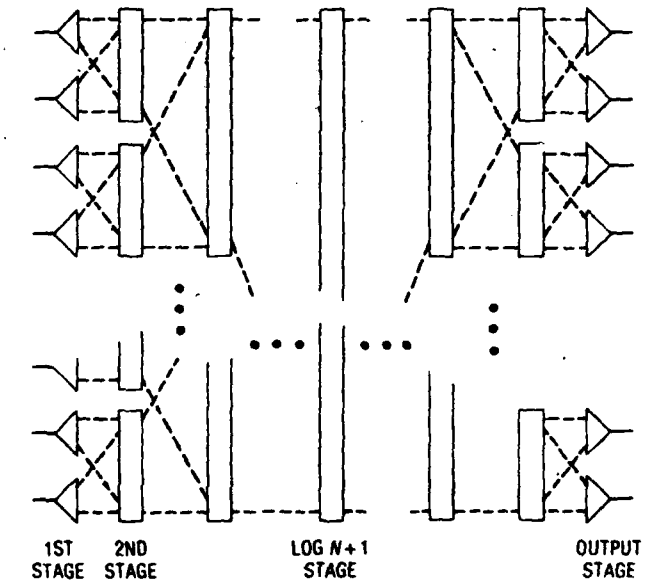


Figure 11. Generalized model of a Cantor network.

The minimal known construction of an (N,N,N)-nonblocking connector has been provided by Cantor.[27,30] Since this construction is considerably different from the other connectors discussed so far, it would be appropriate to sketch it here. Similar to the (N,N,N)-Clos network, Cantor's network has a multiple stage structure which is symmetric around the middle stage. Cantor's approach is to use a particular construction technique and then consider the number of accessible, idle terminals in the middle stage of the resulting network relative to any idle input. This number can be easily calculated in a recursive manner because of the construction technique used. By symmetry, the number is also known relative to any idle output. The sum of the two numbers when compared with the total number of idle terminals in the middle stage at any time indicates that there is a common idle terminal in the middle stage through which the idle input can be connected to the idle output.

To see this in more detail, consider the graph model of a (4,4,4)-Cantor network (shown in Figure 10). Recall that in such a graph model the dotted lines do not represent crosspoints but simply associate nodes which correspond to terminals which would be linked together in a construction. Hence, there are four (1×2)-networks in both the input stage and the output stage; and there are four (2×2)-networks in each of the three intermediate stages. It should be noted also that the association of nodes between stages in the model corresponds to the so-called perfect-shuffle[25,31,32] in a progressing manner. We will now analyze the input side of the network, and by symmetry our observations will also hold for the output side. On the input side, a generalization of network modeled by Figure 11 would indicate that each of the N inputs of the first stage is in a $(1 \times \log N)$-network. This would be followed by $\log N$ stages, each stage consisting of $(N/2)$ $\log N$ (2×2) networks, where the last of these $\log N$ stages (actually the $(\log N + 1)^{th}$ stage in the network) is the middle stage. Again, the perfect shuffle concept is used to progressively associate the nodes in the stages in the model. To further illustrate the above, Figure 12 shows the input side of a (8,8,8)-Cantor network.

As seen from Figure 12 any input is connectable to $\log N$ intermediate terminals which are linked to the inputs of the $\log N$ (2×2)-networks of the second stage. Using these (2×2)-networks in the second

stage, we then see that each of the inputs of the first stage is connectable to $2\log N$ output terminals of the second stage. For a given input, we will call the nodes representing these output terminals of the (in this case) second stage the accessible nodes of that input. The crucial point here is that because of the construction of the network, the number of accessible nodes of an input increases by a factor of two each time we move a stage closer to the middle stage. Hence, when the middle stage is reached, the number of accessible nodes to any input is $N\log N$.

But it is clear that connections that have been previously made and are thereby existing in the network at a given time will limit the actual accessibility of any input to terminals of the middle stage. Hence, we see that our concern must be with *idle*, accessible nodes. Let us then consider the worst case along these lines for stage i, $i=1,\ldots,\log N+1$. We will denote the number of idle, accessible nodes of stage i to any input as $A(i)$. By the construction indicated in Figure 11,

$$A(1) = \log N.$$

Now, the number of accessible nodes on the output side of the second stage is twice that of the output side of the first stage, but note that one of these output nodes can be busy because of other connections being realized in the network. Thus,

$$A(2) = 2A(1) - 1.$$

Similarly,

$$A(3) = 2A(2) - 2.$$

In general,

$$A(i+1) = 2A(i) - 2^{i-1}.$$

Since the initial value is $A(1) = \log N$, the following closed form solutions for $A(i)$ can be obtained for $i \geq 2$:

$$A(i) = 2^{i-1}\log N - (i-1)2^{i-2}.$$

Hence, for any idle input, the number of idle, accessible output nodes of the middle stage is

$$A(\log N + 1) = 2^{\log N}(\log N) - \log N\,(2^{\log N - 1}).$$
$$= \frac{1}{2} N\log N.$$

In the worst case, of the total $N\log N$ terminals on the output side of the middle stage, at least $N\log N - (N-1)$ are idle. Of these idle terminals, at least $(1/2)N\log N$ can be accessed by any idle input. Clearly, in less than worst-case conditions, more of the idle terminals can be accessed. By symmetry a similar argument can be made for any idle output of the network relative to the terminals on the input side of the middle stage. Clearly, this implies that any idle input and any idle output can be interconnected regardless of whatever other interconnections also exist, which is, of course, the desired nonblocking property.

Pippenger[33] has developed an algorithm for finding an interconnecting path through a Cantor network which requires $O(\log N)$ operations.

To determine the number of crosspoints in an (N,N,N)-Cantor network, we note first that the input stage has N $(1\times\log N)$-networks. This results in $N\log N$ crosspoints. Next, we note that the next $\log N - 1$ stages (that is, all the intermediate stages between—and excluding—the input stage and the middle stage) each have $2N\log N$ crosspoints. The same numbers hold for the output side of the middle stage of the network. Summing the number of crosspoints on each side of the middle stage and then adding $2N\log N$ crosspoints for the middle stage yields

$$2(N\log N + (\log N - 1)(2N\log N) + N\log N) =$$
$$4N(\log N)^2 \text{ crosspoints.}$$

If N is a power of 2, then an (N,N,N)-Cantor network has less crosspoints than an (N,N,N)-Clos network for all $N \geq 2^9$. A still further reduction can be made to produce Cantor networks with $O(N(\log N - 1)^2)$ crosspoints and two fewer stages of delay. Recently, Pippenger[29] has slightly improved this bound on the number of crosspoints in a Cantor construction to $16N(\log_5 N)^2$.

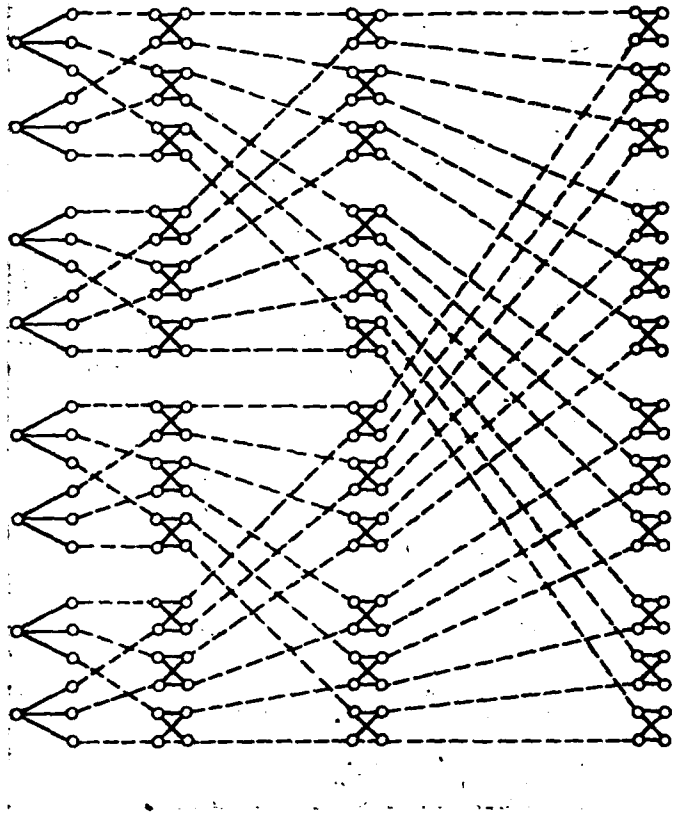


Figure 12. Input side of an (8,8,8)-Cantor network.

Partitioners

A partitioner is a circuit switching network which performs the function of partitioning a set of devices usually attached to the inputs of the network into disjoint subsets so that the devices within each subset can communicate with each other over private buses. Clearly, the most straightforward construction of a partitioner is an $(N\times(N/2))$-network where the devices to be partitioned are attached to the N inputs. Notice that if the outputs of the network are indistinguishable in their use, the $N/2$ horizontal lines of such a network can be viewed as simply $N/2$ indistinguishable buses to be shared by the subsets of devices. Then the function of the $(N\times(N/2))$-network would simply be to interconnect each of the specified subsets of input devices to a distinct bus line, where the specification of the input subsets could either be according to size (if the input devices are indistinguishable) or according to an exact listing of the input devices (if the input devices are distinguishable). Notice also that if the outputs were attached to distinguishable output devices so that subsets of input devices had to be connected to particular buses in the system operation, the $(N\times(N/2))$-network could also realize this type of partitioning demand.

Such a complete crossbar circuit switch realization of a partitioner, however, requires $O(N^2)$ crosspoints. Hence, in spite of the simplicity of control, for most systems which demand partitioning capabilities, alternative designs which require fewer crosspoints must be considered. Along these lines, in addition to the other issues with which we have become involved in our consideration of circuit switching networks, two further issues must be considered for partitioners: we must establish whether or not the devices to be partitioned are identical; we must establish whether or not the outputs of the partitioner are to be specified.

If the system consists of identical devices, the initial formation of subsets is greatly simplified because our concern is only with interconnecting unspecified groups of specified sizes. If the devices are distinguishable, for example, the devices might be a collection of processors and memories; or if the devices are physically identical but are used in the processing of concurrent, asynchronous tasks so that their states make them functionally distinguishable, then the partitioning is clearly more demanding.

Similarly, as suggested above, the outputs must sometimes be specified in a partitioning request. For example, each such output can be a port to some specific device or function, or it could be a signal or data source required by one of the subsets of devices in the partition. Such cases are clearly more complicated to realize than those in which the outputs are indistinguishable.

On the basis of the above, we will refer in the following to (N distinguishable inputs, M distinguishable outputs)-partitioners as (ND,MD)-partitioners; (N distinguishable inputs, M indistinguishable outputs)-partitioners as (ND,MI)-partitioners; (N indistinguishable inputs, M distinguishable outputs)-partitioners as (NI,MD)-partitioners; and (N indistinguishable inputs, M indistinguishable outputs)-partitioners as (NI,MI)-partitioners. Furthermore, if all partitions cannot be realized by a partitioner, we will indicate this by referring to the partitioner as incomplete. Partitioning can also be performed in a nonblocking or in a rearrangeable manner. Nonblocking partitioning involves the capability of establishing the requested interconnection of a subset of devices without disturbing the existing subset interconnections presently implemented by the network. Rearrangeability implies that such disturbances occur. Indeed, not only must paths be rearranged, but at times the devices composing currently interconnected subsets must also be rearranged. Such classification of networks will not be stressed in the following. However, in some of the suggested approaches to the realization of partitioning demands, the use of, for example, connectors will be suggested; it will be apparent, however, that even if, say, nonblocking connectors were used in those situations, nonblocking partitioning will not always result.

The simplest partitioning requirement clearly involves indistinguishable inputs and indistinguishable outputs. In fact, for such a requirement and N devices, a straightforward serial structure with $N-1$ crosspoints as shown explictly in Figure 13(a) is sufficient. Note that we take advantage of the fact that the outputs are indistinguishable to design a network with effectively no outputs. Still simpler structures are possible. For example, if $N=5$, all seven possible

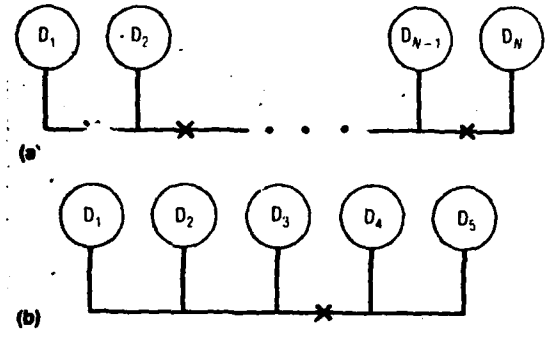


Figure 13. Partitioning networks with indistinguishable inputs and outputs.

Table 1.
Partitions of five identical elements.

Partitions of Devices	Number of Subsystems
((D), (D), (D), (D), (D))	5
((D,D), (D), (D), (D))	4
((D,D), (D,D), (D))	3
((D,D,D,), (D), (D))	3
((D,D,D), (D,D))	2
((D,D,D,D), (D))	2
(D,D,D,D,D)	1

partitions are listed in Table 1. Then with one crosspoint and the hardwired, fixed links as shown in Figure 13(b), all such partitions can be realized under the assumption that each of the devices can be placed in a subset of size 1 by the action of avoiding or ignoring communication with other devices to which it might be hardwired by means of a fixed link.

The case of $(N\mathrm{I},M\mathrm{D})$-partitioning adds a certain degree of complexity to the requirements to be satisfied. First of all, given N input devices and $M=N$ outputs, it can be seen that the total number of partitions which can be requested is

$$\binom{2N-1}{N} = \frac{1}{2}\,\frac{(2N)!}{(N!)^2} .$$

With Stirling's formula,[18] it can then be shown that

$$\binom{2N-1}{N} \simeq \frac{1}{2}(N\pi)^{1/2}\,(2^{2N}) .$$

So, theoretically, $O(N)$ crosspoints are required for the realization of an $(N\mathrm{I},M\mathrm{D})$-partitioner.

However, other than a complete crossbar network, the only construction which the authors are aware of requires $O(N\log N)$ crosspoints. As illustrated in Figure 14, this consists of an (N,N,N)-connector, which we have seen can be realized with $O(N\log N)$ crosspoints, augmented by a series of $N-1$ crosspoints serially linking the inputs. The operation of this network in providing a set of, say, k input devices to some specified output is to simply interconnect a set of k consecutive input devices, and then provide a path through the (N,N,N)-connector from any one of the inputs associated with this set to the specified output. if the (N,N,N)-connector is a Waksman network,[21] the resulting $(N\mathrm{I},M\mathrm{D})$-partitioner has $4N\log N-3N+3$ crosspoints. A similar scheme is possible with Benes' $3.8N\log N$ crosspoint network.

The case of $(N\mathrm{D},M\mathrm{I})$-partitioners has received considerable attention in the literature. Goke and Lipovski[34] have described a construction for such partitioners, which they have referred to as *banyan networks*. Figure 15 shows an example of the graph model of an (8D,8I)-banyan partitioner. In general, the graph model of a banyan network is a Hasse diagram of partial order in which there is an unique path from any base to the apex, where a base (apex) is a node with no edges incident into (out from) it.

In this figure, the bottom nodes, which represent inputs that are assumed to be attached to the devices to be partitioned, are the bases and the top nodes are the apexes. Because of the generality of its definition, many types of networks are included in the class of banyan networks. But those banyan networks which are of practical interest do not usually have full partitioning capabilities, so that Goke and Lipovski considered parallel and multiplexed usage of banyan networks to obtain a full partitioning capability. Some of the motivations for considering banyans are that such networks have $O(N\log N)$ crosspoints and only $O(\log N)$ delay. In addition, configuring the banyan network to satisfy a device subset interconnection request is straightforward and can be accomplished in $O(\log N)$ operations. Finally, given an existing state of a banyan network, if a new request for a subset interconnection is made by a subset of the idle input devices, and if the resulting partition consisting of the current realized subset interconnections plus the new subset interconnection request is realizable by the network (recall that the banyan network is an incomplete partitioner), then the banyan can provide the new subset interconnection request without disturbing the existing subset interconnections.

Recently, Thompson[13] has suggested the use of his expanders as $(N\mathrm{D},N\mathrm{I})$-partitioners with full partitioning capability. An expander is similar to a connector except that it has the added interconnection capability of allowing inputs to be interconnected to any number of outputs, while still requiring that each output be interconnected to at most one input. Hence, just as an (N,N,N)-connector can realize all $N!$

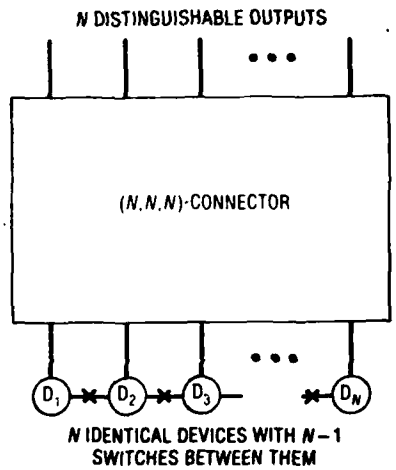


Figure 14. $(N\mathrm{I},N\mathrm{D})$ partitioner.

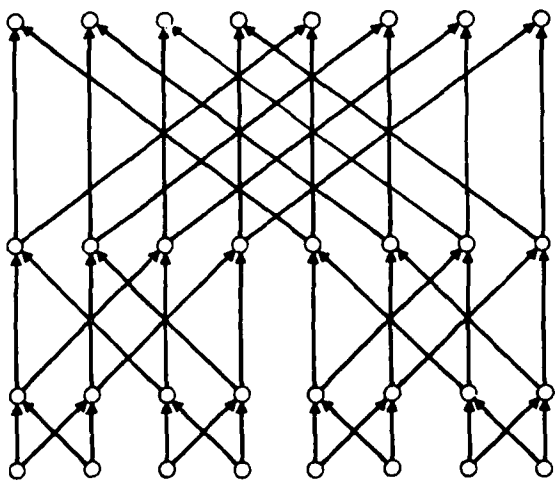


Figure 15. (8D,8I)-banyan partitioner.

permutation mappings of the inputs onto the inputs, then an (N,N,N)-expander can realize all N^N general mappings of the inputs to the outputs. Thompson's design is to simply attach the devices to be partitioned to the outputs of one of his expanders. Since the inputs of his expander (the outputs of the partitioner) are now indistinguishable, as far as the partitioner's operation is concerned, a part of the input side of the expander can be deleted, resulting in an $(ND, (N/2)I)$-partitioner with $6N\log N$ crosspoints. Slight improvements to this partitioning scheme are possible by using the (N,N,N)-Waksman connection as part of Thompson's expander or by using an (N,N,N)-Benes connector[1] where N is a power of 3 and three-way branching is used throughout.

It is also possible to use the (N,N,N)-connector augumented by $N-1$ serial crosspoints on the output side for full partitioning capability. This is simply the reverse of the network proposed for the (NI,ND)-partitioner. Such an (ND,NI)-partitioner is illustrated in Figure 16. Such complete partitioners require $4N\log N-3N+3$ crosspoints, but they do not provide the nonblocking property. Again, a similar scheme is possible with Benes' $3.8N\log N$ crosspoint network.

It is also possible to construct a partitioner from an $O(N\log N)$ crosspoint (N,N,N)-connector with feedback.[34] A clear disadvantage of this design of a partitioner is that the paths which serve as buses for subset interconnection iterate around and through the network, introducing significant delays. However, with an (N,N,N)-Cantor network, nonblocking partitioning can be achieved.

Finally, it should be clear that (ND,MD)-partitioners can be realized with (M,N,N)-expanders, where the devices to be partitioned are attached to the outputs of the expander.

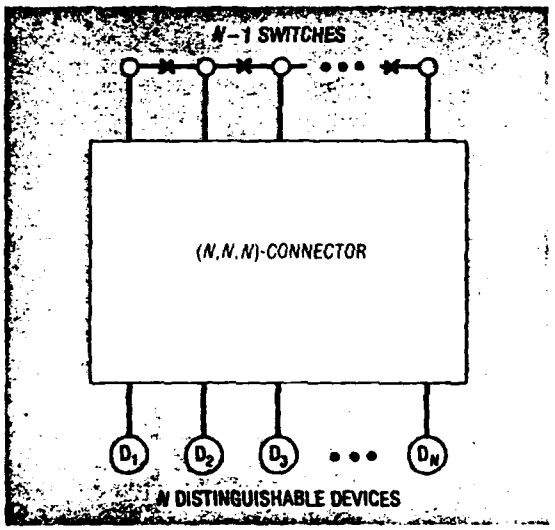


Figure 16. Complete (ND, NI)-partitioner with $4N\log N - 3N + 1$ crosspoints.

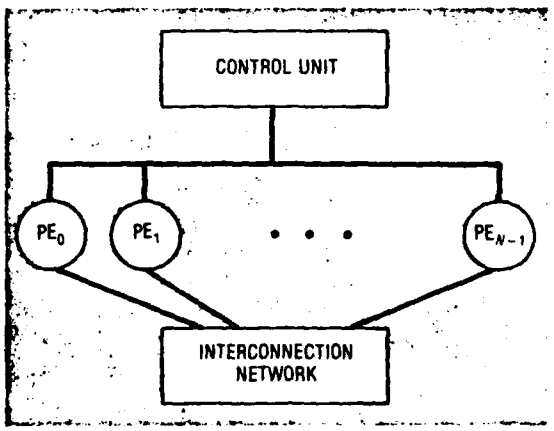


Figure 17. Schematic view of an SIMD computer.

SIMD interconnectors, expanders, sorters . . .

There are still large, open areas in our circuit switching sampler; but to fill them in would go far beyond our page limitations. Accordingly, we will conclude by mentioning just a few of the many other possible designs.

For data exchanges between processing elements in classes of computer systems having structures referred to as *single-instruction multiple-data-stream architectures*, a number of interesting interconnection patterns have been developed. Figure 17 shows a schematic view of an SIMD-computer. The interconnection network shown is not necessarily a separate functional unit in an actual SIMD computer, but the implementation of interconnection patterns is a fundamental aspect of SIMD-computer operation. Accordingly, these interconnection patterns can be described in terms of single-stage interconnection networks containing from $O(N)$ to $O(N\log N)$ crosspoints. These networks will be referred to as SIMD-interconnectors.

The N inputs and the N outputs of the SIMD-interconnectors can be considered to be attached to the N processing elements so that, say, processing element i can transmit data to the processing elements $i_{j_1},\ldots,i_{j_k}$ if input i has crosspoints between it and outputs $j_1,\ldots,j_k$. Since no input of the SIMD-interconnector will have crosspoints between it and all the outputs, to perform a data exchange between specified pairs of processing elements, a control unit must place the SIMD-interconnector in a sequence of states so that the desired data exchange is accomplished by a series of data exchanges through intermediate processing elements. Clearly, for SIMD-interconnectors, complexity of the control algorithm and the time delay required for data exchanges are of major significance.

The general states which the SIMD-interconnector can assume are usually described by functions which map the inputs onto the outputs. However, whether or not a specific data exchange indicated by a function actually occurs in system operation depends

upon whether the processing elements involved are in active or inactive states. This is determined by a masking scheme. As for the details of these masking schemes, suffice it to say that from our perspective, an inactive processing element implies that the crosspoints between it and the outputs in the SIMD-interconnector which are involved in the realization of some function are considered to be open (that is, no data is transmitted), whereas those crosspoints relative to an active processing element are considered to be closed (that is, data is transmitted). Hence, the complete data exchange indicated by a function does not occur.

There are five principal *SIMD-interconnectors* which have been discussed in the literature. These five are all closely related; in fact, Siegel[35] has shown that they are all effectively equivalent in the sense that each can simulate the interconnecting capabilities of the others. Since SIMD-interconnectors are discussed in a companion paper in this issue of *Computer*, we will only reference, for completeness, these five principal networks here: the shuffle-exchange network,[25,31,32,36] which has $2N$ crosspoints; the cube network,[37] which has $N\log N$ crosspoints; the Illiac network[36,38-41] which has $4N$ crosspoints; the plus-minus 2^k network, [36,42,43] which has $O(N\log N)$ crosspoints; and the wraparound plus-minus 2^k network, which has $O(N\log N)$ crosspoints.

Expanders (also referred to as generalized connection networks) have already been mentioned as circuit switching networks which realize (assuming there are N inputs and N outputs) all N^N mappings of the inputs onto the outputs. In other words, one-to-many assignments are permitted and the expanders have a fanout capability. Such networks are more powerful than connectors which realize all $N!$ one-to-one mappings. Indeed, it would be possible here to present for expanders a section analogous to that on connectors; unfortunately, only the following very brief sketch is possible. Ofman[14] originally showed the existence of a $10N\log N$ crosspoint expander with $5\log N$ delay; Pippenger improved this result by showing the existence of a $7.6N\log N$ crossppoint expander with an $O(N\log N)$ delay; Thompson[13] then showed the existence of the same size network with only $3.8\log N$ delay. Recently, Pippenger,[44] utilizing his existence results for concentrators and superconcentrators, placed a lower bound of $1.9N\log N$ crosspoints and an upper bound of $3.8N\log N$ crosspoints on expanders with $O((\log N)^2)$ delays. Regarding constructions, Masson and Jordan[45] have shown explicit designs for $O(N^{5/3})$ crosspoint networks with only three delays. Repeated decompositions using this design result in rearrangeable expanders with $O(N^{3/2}\log N)$ crosspoints and nonblocking expanders with $O(N^{3/2}(\log N)^{\log 6/\log 3})$ crosspoints. Further work along these lines has been done by Masson,[46] Hwang,[47] and Asthana.[48] Pippenger[33] gives a non-uniform rearrangeable expander using feedback. It is simply constructed from two (N,N,N)-connectors, one used specifically for feedback. Hence, an interconnecting path involving one input and many (perhaps all) outputs can iterate

through the network up to $O(N\log N)$ times. Clearly, the paths through such a network are non-uniform, but by the work of Tsao-Wu and Opferman,[22] a state realizing a total assignment can be determined with $O(N\log N)$ operations. Finally, Masson[5] has also shown some expander constructions by using stages of the binomial concentrators. Pippenger[33] uses a similar idea to show that $O(N(\log N)^{v+1})$ crosspoint expanders can be constructed if $O(N(\log N)^{v})$ crosspoint concentrators are used. Since a connector certainly operates as a concentrator, we can therefore use this technique to construct an $O(N(\log N)^3)$ crosspoint expander by using Cantor's $O(N(\log N)^2)$ crosspoint connectors. Similarly, this approach shows that if a state satisfying a new request can be determined with $O((\log N)^{w})$ operations for the concentrators, then the resulting expander requires $O((\log N)^{w+1})$ operations to do the same.

Sorting is such a basic operation and so similar in nature to connecting, that as might be expected, sorting networks, or sorters, represent an area unto themselves. The primary difference between a sorter and a connector is that the former is made up of comparater cells, which for our purposes can be considered to be analogous to (2×2)-networks. Hence the control of sorters can be thought of as being distributed to each such cell. Also like nonblocking connectors, there is no reason not to expect $O(N\log N)$ crosspoint designs; but, again, the minimal known construction of a sorter, which utilizes an interesting odd-even merging concept, is the Batcher sorter[49] which has $O(N(\log N)^2)$ crosspoints. However, it has been shown[4] that the Batcher sorter is not minimal for N=9,10,12, and 16, which places in suspicion the minimality of the Batcher sorter for arbitrary N. Marcus[3,50] observes and discusses the interesting difference in the $O(N\log N)$ growth of the computation time of software programs that sort as compared to this $O(N(\log N)^2)$ growth of crosspoints in sorters, and attributes the discrepancy to the adaptive nature of the former as opposed to the fixed structure of the latter. Other significant sorter designs and studies are due to Bose and Nelson[51] and Van Voorhis[52-55]; some of Van Voorhis' work also yields sorter designs with the same order of growth but yet fewer crosspoints than the Batcher sorter.

And there are many other circuit switching networks and issues which our sampler could display: cellular interconnection arrays[56,57]; programmable indexing networks[58]; seldom blocking networks and connectors with finite blocking probability[50,59]; time-division circuit switching networks and time-slot interchangers[50,60-62]; and, alas, the list could go on and on. We will cease our presentation here with the following comments. Much of the work which now represents the state of the circuit switching area was motivated by general communications-oriented problems; indeed, this still represents much of the motivation for research currently in progress. However, a new influence on such research—which some say will eventually be *the* influence—is the development of LSI/VLSI devices to the point where very large networks of such devices are practical, perhaps even unavoidable. Some type of interconnector chip could eventually be as common in such system designs as processor chips, memory chips, and I/O chips are today. Before such a point is reached, however, many questions need to be addressed, for, clearly, the applicability of a circuit switching network to a system design is related to the details of the application. For example, what are the generic circuit switching network requirements for, say, various types of fault-tolerant computing designs as opposed to that required for, say, various types of vector processing arrays? Some preliminary investigations along these lines have been reported [63-68]; clearly, however, there is a long way to go. Moreover, the treatment of such questions seems to demand an intriguing blend of computer science and computer engineering research; for example, complexity theory and digital systems structures are intimately related to such considerations. It is an exciting challenge, and one that cannot be ignored. It is only necessary to scan the modern literature describing telecommunication, computer/peripheral, or instrumentation complexes to see this, for inevitably in a schematic or illustration there will be that operationally significant box labeled "circuit switching network." ■

References

1. V. Beneš, *Mathematical Theory of Connecting Networks and Telephone Traffic*, Academic Press, New York, 1965.
2. K. Thurber, "Interconnection Networks—A Survey and Assessment," *AFIPS Conf. Proc. 1974 NCC*, Vol. 43, pp. 909-919.
3. M. Marcus, "The Theory of Connecting Networks and Their Complexity: A Review," *Proceedings of the IEEE*, Vol. 65, No. 9, Sept. 1977, pp. 1263-1271.
4. D. Knuth, *The Art of Computer Programming, Vol. III: Sorting and Searching*, Addison-Wesley, Reading, Mass, 1973.
5. G. Masson, "Binomial Switching Networks for Concentration and Distribution," *IEEE Trans. Comm.*, Vol. COM-25, No. 9, Sept. 1977, pp. 873-883.
6. M. Pinsker, "On the Complexity of a Concentrator," *Proc. 75th International Teletraffic Conference*, Stockholm, 1973, pp. 318/1-318/4.
7. N. Pippenger, "On the Complexity of Strictly Nonblocking Concentration Networks," *IEEE Trans. Comm.*, Vol. COM-22, 1974, pp. 1890-1892.
8. G. Margulis, "Explicit Construction of Concentrators," translated from *Problemy Peredachi Informatsii*, Vol. 9, No. 4, 1973, pp. 71-80, and published in *Problems in Information Transmissions*, Plenum Publishing Corp., New York., 1975, pp. 325-332.
9. G. Masson, G. Gingher, and S. Nakamura, "Variations in Binomial Concentrator Designs," in preparation.
10. L. Valiant, "On Non-linear Lower Bounds in Computational Complexity," *Proc. 7th Ann. ACM Symp. on Theory of Computing*, Albuquerque, N.M., 1975.
11. N. Pippenger, "Superconcentrators," *SIAM Jour. Comp.*, Vol. 6, No. 2, Feb. 1977, pp. 298-304.

12. P. Erdos and J. Spencer, *Probabilistic Methods in Combinatorics*, Academic Press, New York, 1974.

13. C. Thompson, "Generalized Connection Networks for Parallel Processor Intercommunication," *IEEE Trans. Comput.*, Vol. C-27, No. 12, Dec. 1978, pp. 1119-1125.

14. J. P. Ofman, "A Universal Automaton," *Trans. Moscow Math. Soc.*, Vol. 14, 1965 (translation published by Amer. Math. Soc., Providence, R.I., 1967, pp. 200-215.

15. C. Shannon, "Memory Requirements in a Telephone Exchange," *Bell Systems Tech. J.*, Vol. 29, 1950, pp. 343-349.

16. W. Feller, *An Introduction to Probability Theory and Its Applications*, Vol. 1, John Wiley and Sons, Inc., New York, 3rd ed. 1968.

17. P. Hall, "On Representatives of Subsets," *J. London Math. Soc.*, Vol. 10, pp. 26-30, 1935.

18. M. Paull, "Reswitching of Connection Networks," *Bell System Tech. J.*, Vol. 41, No. 3, May 1962, pp. 833-855.

19. B. Beizer, "The Analysis and Synthesis of Switching Networks," *Proceedings of Symposium on Math. Theory of Automata*, 1962, pp. 563-576.

20. A. Joel, Jr., "On Permutation Switching Networks," *Bell System Tech. J.*, Vol. 47, No. 5, May-June 1968, pp. 813-822.

21. A. Waksman, "A Permutation Network," *JACM*, Vol. 15, No. 1, Jan. 1968, pp. 159-163.

22. D. C. Opferman and N. T. Tsao-Wu, "On a Class of Rearrangeable Switching Network, Part I: Control Algorithms," Part II: Enumeration Studies and Fault Diagnosis," *Bell System Tech. J.*, Vol. 50, No. 5, May-June 1971, pp. 1579-1618.

23. J. Lenfant, "Parallel Permutations of Data: A Benes Network Control Algorithm for Frequently Used Byections," *IEEE Trans. Comput.*, Vol. C-27, No. 7, July 1978, pp. 637-647.

24. D. H. Lawrie, "Access and Alignment of Data in an Array Processor," *IEEE Trans. Comput.*, Vol. C-24, No. 12, Dec. 1975, pp. 1145-1155.

25. H. S. Stone, "Parallel Processing with the Perfect Shuffle," *IEEE Trans. Comput.*, Vol. C-20, No. 2, Feb. 1971, pp. 153-161.

26. C. Clos, "A Study of Non-Blocking Switching Networks," *Bell System Tech. J.*, Vol. 32, No. 2, Mar. 1953, pp. 406-424.

27. D. G. Cantor, "On Non-Blocking Switching Networks," *Networks*, Vol. 1, No. 4, Winter 1971, pp. 367-377.

28. L. A. Bassalygo and M. S. Pinsker, "On the Complexity of Optimal Non-Blocking Switching Networks Without Rearrangement," in *Problems in Information Transmission*, Plenum Publishing Corp., New York, 1973, pp. 84-87.

29. N. Pippenger, "On Rearrangeable and Non-Blocking Switching Networks," *Journal of Computer and System Science*, Vol. 17, No. 4, Sept. 1978, pp. 145-162.

30. D. Cantor, "On Construction of Nonblocking Switching Networks," *Proc. of Symposium on Computer-Communications Networks and Teletraffic*, Polytechnic Institute of Brooklyn, 1972.

31. S. W. Golomb, "Permutations by Cutting and Shuffling," *SIAM Review*, Vol. 3, No. 4, Apr. 1961, pp. 293-297.

32. P. Johnson, "Congruences and Card Shuffling," *American Mathematical Monthly*, Vol. 63, No. 10, Dec. 1956, pp. 718-719.

33. N. Pippenger, "The Complexity Theory of Switching Networks," MIT Res. Lab. of Electronics, Rep. TR-487, 1973.

34. L. Goke and G. Lipovski, "Banyan Networks for Partitioning Multiprocessor Systems," *Proc. 1st Ann. Comput. Architecture Conf.*, 1973, pp. 21-28.

35. H. Siegel, "Single Instruction Stream—Multiple Data Stream Machine Interconnection Network Design," *Proc. 1976 Int. Conf. on Parallel Processing*, 1976, pp. 273-282.

36. H. J. Siegel, "Analysis Techniques for SIMD Machine Interconnection Networks and the Effects of Processor Address Masks," *Proc. 1975 Sagamore Conference on Parallel Processing*, pp. 106-109.

37. K. E. Batcher, "The Multi-Dimensional Access Memory in STARAN," *IEEE Trans. Comput.*, Vol. C-26, No. 2, Feb. 1977, pp. 174-177.

38. G. H. Barnes, R. M. Brown, M. Kato, D. J. Kuck, D. L. Slotnick, and R. A. Stokes, "The ILLIAC IV Computer," *IEEE Trans. Comput.*, Vol. C-17, No. 8, Aug. 1968, pp. 746-757.

39. W. J. Bouknight, S. A. Denenberg, D. E. McIntyre, J. M. Randall, A. H. Sameh, and D. L. Slotnick, "The ILLIAC IV System," *Proceedings of the IEEE*, Vol. 60, No. 4, Apr. 1972, pp. 369-388.

40. S. E. Orcutt, "Computer Organization and Algorithms for Very-High Speed Computation," Dept. of Computer Science, Stanford University, *Ph.D. Thesis*, 1974.

41. D. L. Slotnick, W. C. Borck, and R. C. McReynolds, "The SOLOMON Computer," *AFIPS Conf. Proc., FJCC*, Vol. 22, 1962, pp. 97-107.

42. T. Feng, "Data Manipulating Functions in Parallel Processors and Their Implementations," *IEEE Trans. Comput.*, Vol. C-23, No. 3, Mar. 1974, pp. 309-318.

43. T. Feng, "Parallel Processing Characteristics and Implementation of Data Manipulating Functions," Dept. of Electrical and Computer Engineering, Syracuse University, RADC-TR-73, 1973.

44. N. Pippenger, "Generalized Connectors," *SIAM Jour. Comp.*, Vol. 7, No. 4, Apr. 1978, pp. 510-514.

45. G. Masson and B. Jordan, "Generalized Multi-stage Connection Networks," *Networks*, Vol. 2, No. 3, Fall 1972, pp. 191-209.

46. G. Masson, "Upper Bounds on Fanout in Connection Networks," *IEEE Trans. Circuit Theory*, Vol. CT-20, No. 3, May 1973, pp. 222-230.

47. F. Hwang, "Rearrangeability of Multi-connection Three Stage Clos Networks," *Networks*, Vol. 2, No. 4, Winter 1972, pp. 301-306.

48. A. Asthana, "Design and Control of a Three-State Switch Matrix in the Presence of Fan-Out," *IEEE Trans. Comput., Vol. C-27, No. 10, Oct. 1978*, pp. 886-895.

49. K. Batcher, "Sorting Networks and Their Applications," *AFIPS Conf. Proc., 1968 SJCC*, pp. 307-314.

50. M. Marcus, "New Approaches to the Analysis of Connecting and Sorting Networks," MIT Res. Lab. of Electronics, Rep. TR-486, 1972.

51. R. Bose and R. Nelson, "A Sorting Problem," *JACM*, Vol. 9, No. 2, Apr. 1962, pp. 282-296.

52. D. Van Voorhis, "A Generalization of the Divide-Sort-Merge Strategy for Storting Networks," Stanford Electronics Laboratories, TR-16, 1971.

53. D. Van Voorhis, "Large [g,d] Sorting Networks," Stanford Electronics Laboratories, TR-18, 1971.

54. D. Van Voorhis, "Toward a Lower Bound for Sorting Networks," in *Complexity of Computer Computations*, Plenum Press, 1972.

55. D. Van Voorhis, "An Improved Lower Bound for Sorting Networks," *IEEE Trans. Comput.*, Vol. C-21, No. 6, June 1972, pp. 612-613.

56. W. Kautz, K. Levitt, and A. Waksman, "Cellular Interconnection Arrays," *IEEE Trans. Comput.*, Vol. C-17, No. 5, May 1968, pp. 443-451.

57. S. Bandyopadhyay, S. Basu, and A. Choudhurg, "A Cellular Permuter Array," *IEEE Trans. Comput.*, Vol. C-21, No. 10, Oct. 1972, pp. 1116-1119.

58. K. Thurber, "Programmable Indexing Networks," *AFIPS Conf. Proc., 1970 SJCC*, pp. 51-58.

59. N. Pippenger, "The Complexity of Seldom-Blocking Networks," *Proc. 1976 Int. Conf. on Communications*, pp. (7-8)-(7-12).

60. M. Marcus and H. McDonald, "The Queuing Crossbar: A Hybrid Time-Division and Space-Division Switching Network," *Proc. 1969 National Electronics Conf.*, pp. 605-610.

61. M. Marcus, "Space-Time Equivalents in Connecting Networks," *Proc. 1970 International Conf. on Communications*, pp. (35-25)-(35-31).

62. M. Marcus, "Designs for Time-Slot Interchangers," *Proc. 1970 National Electronics Conf.*, pp. 812-817.

63. W. Kautz, "The Design of Optimal Interconnection Networks for Multiprocessor Applications," *Architecture and Design of Digital Computers*, NATO Advanced Summer Institute, 1969.

64. K. Levitt and W. Kautz, "Cellular Arrays for the Solutions of Graph Problems," *CACM*, Vol. 15, No. 9, Sept. 1972, pp. 789-801.

65. K. Levitt, M. Green, and J. Goldberg, "A Study of Data Communication Problems in a Self-Repairable Multiprocessor," *AFIPS Conf. Proc., 1968 SJCC*, pp. 515-527.

66. J. Goldberg, K. Levitt, and J. Wensley, "An Organization for a Highly Survivable Memory," *IEEE Trans. Comput.*, Vol. C-23, No. 7, July 1974, pp. 693-706.

67. K. Thurber and G. Masson, "Recent Advances in Microprocessor Technology and their Impact on Interconnection Design in Computer Systems," *Proc. 1977 International Conf. on Communications*, pp. (46.2-216)-(46.2-220).

68. K. Thurber and G. Masson, *Distributed Processor Communication Architectures*, D.C. Heath and Company, 1979.

Gerald M. Masson is an associate professor of electrical engineering at The Johns Hopkins University. His research interests include fault-tolerant computing, interconnection networks, and computer systems design. He received the BSEE degree in 1966 from the Illinois Institute of Technology and the MS and PhD degrees in electrical engineering from Northwestern University in 1968 and 1971, respectively.

George C. Gingher is a control engineer for Bethlehem Steel, where he has been employed for over 20 years. He is a PhD candidate in the Electrical Engineering Department at Johns Hopkins University.

Gingher received the BSEE in 1963 and the MS in applied mathematics, both from Johns Hopkins.

Shinji Nakamura is a PhD candidate in the Electrical Engineering Department at Johns Hopkins. He received a BS and an MS in Physics in 1966 and 1969, respectively, from Gakushuin University, in Tokyo. From 1969-76 he was employed by the Mitsubishi Electric Company. He also received an MS degree in computer science from the University of Illinois at Champaign-Urbana in 1976 and the MSEE from The Johns Hopkins University in 1977.

CAPACITY CALCULATION OF COMPOSITE CONCENTRATORS

SHINJI NAKAMURA AND GERALD M. MASSON
Department of Electrical Engineering
The Johns Hopkins University
Baltimore, Maryland 21218

ABSTRACT

In this paper, a capacity calculation technique is described for a class of interconnection networks called composite concentrators. This technique is then used to develop some capacity observations for various concentrators of this class.

INTRODUCTION

A concentrator is a type of interconnection network having switching elements, called crosspoints, between disjoint sets of inputs and outputs to the network such that specified sets of inputs of some maximum size can be interconnected to arbitrary sets of outputs of the same size [1]. More specifically, a concentrator with N inputs and M outputs, $N > M$, is said to have a capacity $c \leq M$ if, for any choice of $K \leq c$ inputs, a set of K disjoint paths from the inputs can be established by closing crosspoints to some K outputs. It is important to note that the inputs can be specified, but the outputs to which they are connected cannot be specified. Clearly, if crosspoints were placed between every input and output, the capacity would be equal to the number of outputs. (Indeed, for such a network, the outputs could also be specified.) However, in general, the number of crosspoints would be prohibitively high; hence, sparser constructions are of interest. The problem with which we deal in this paper is the capacity calculation of a class of one-stage sparse crosspoint networks called composite concentrators.

Preliminaries

Figure 1 shows three examples of one-stage (or bipartite) sparse crosspoint networks. Such networks can be expressed formally as a triplet (I,O,R), where I is the set of inputs, O is the set of outputs, and R is a relation detailing the crosspoint placement between the inputs and the outputs. R can be given in the form of a graph, or as a crosspoint model (as is done in Figure 1), or as a set of ordered pairs where $(i,j) \in R$ if there is a crosspoint between input i and output j. Regarding the latter, we can express the network of Figure 1(a) as

$$(\{1,2,3,4,5\}, \{a,b,c\}, \{(1,a),(1,c),(2,b),(3,c),(4,b),(4,c),(5,a),(5,b)\}) .$$

Similarly, the networks of Figures 1(b) and 1(c) can respectively be expressed as

$$(\{1,2,3,4,5,6\},\{a,b,c,d\},\{(1,a),(1,b),(2,a),(2,c),(3,a),(3,d),(4,b),(4,c),(5,b),(5,d),(6,c),(6,d)\})$$

and

$$(\{1,2,3\},\{a,b,c\},\{(1,a),(1,b),(2,a),(2,c),(3,b),(3,c)\}) .$$

It should be noted that the networks of Figures 1(a) and 1(b) satisfy the implicit requirement for concentrators that the number of inputs is greater than the number of outputs and (since the capacity must be greater than or equal to one and all outputs must be of some use in providing interconnecting paths from the inputs through the network) that each input (output) can potentially be connected to at least one output (input). More formally, we can express one-stage concentrators as triplets (I,O,u) where

$|I| > |O|$, and where u (u^{-1}) is a relation mapping elements of $I(O)$ to subsets of $O(I)$ corresponding to the crosspoint placement such that $u(i) \neq \phi$ ($u^{-1}(o) \neq \phi$) for all $i \in I$ ($o \in O$).

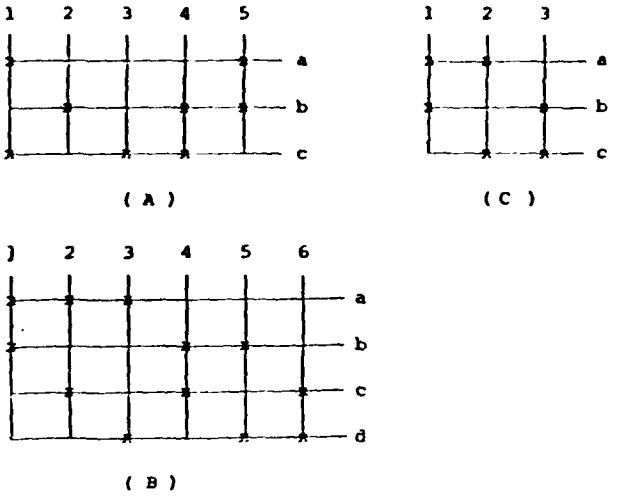


Figure 1

It should also be pointed out for later reference that Figure 1(b) is a basic binomial concentrator which we will describe as a $\binom{4}{2}$-concentrator since the crosspoint pattern connecting the outputs to the inputs consists of all possible $\binom{4}{2}$ choices of 2 crosspoints between the inputs and the 4 outputs. In general, an $\binom{x}{y}$-concentrator has $\binom{x}{y}$-inputs, x-outputs, $y \cdot \binom{x}{y}$ crosspoints. It should also be noted that the network of Figure 1(c) is $\binom{3}{2}$-network, but it is not a concentrator.

The concentrator of Figure 1(a) has a capacity of 2 and the $\binom{4}{2}$-concentrator of Figure 1(b) has a capacity of 4. However, the capacity of a sparse crosspoint networks which satisfies the implicit requirements for a concentrator is, in general, very difficult to determine. Clearly, one way to determine the capacity is to exhaustively consider subsets of inputs (starting with subsets of size two and progressing upwards) until a subset is found for which there is not a one-to-one mapping of the inputs to some subset of the outputs. However, even for networks of modest size this is prohibitive. Hence, in this paper we will develop a much more efficient capacity calculation technique with which to address this problem for a class of concentrators called composite concentrators.

Isolation Concepts

To develop our capacity calculation scheme it is necessary that we first define the following concepts.

For a concentrator expressed as (I,O,u), if $u(i) \subseteq O' \subseteq O$ for some $i \in I$, then we will say that i is completely enclosed by O', or O' isolates i. For a given subset $O' \subseteq O$, the set $\sigma(O')$ is:

$$\sigma(O') = \{i \mid i \in I \text{ and } u(i) \subseteq O'\}.$$

$\sigma(O')$ is called the isolated set of O', since elements of $\sigma(O')$ cannot access any output except those which are the elements of O'.

For all subsets O' of O for which $|O'| = k$, we define an isolation

number H_k to be:

$$H_k = \max_{|O'|=k \text{ and } O' \subseteq O} |\sigma(O')| .$$

Finally, the isolation vector, $(h_1,h_2,\ldots,h_{|O|})$, is determined by taking the differences of the isolation numbers as follows:

$$h_j = H_j - H_{j-1} \quad \text{where } 1 \leq j \leq |O| \text{ and } H_0 = 0 .$$

Obviously $\sum_{j=1}^{k} h_j = H_k$, and $\sum_{j=1}^{|O|} h_j = H_{|O|} = |I|$.

Example 1: Consider the concentrator of Figure 1(a). By inspecting the relation u it is seen that {a,b} isolates {2,5} , because $u(2)=\{b\} \subset \{a,b\}$. and $u(5)=\{a,b\} \subset \{a,b\}$.
Since $|u(2)|=|u(3)|=1$ but $u(2) \neq u(3)$ and $|u(j)|>1$ for $j \in \{1,4,5\}$, it follows that $H_1=1$. Similarly $H_2=3$, and $H_3=5=|I|$. Hence the isolation vector of this concentrator is $(h_1,h_2,h_3)=(1,2,2)$.

Capacity Calculation

We can now give the following:

Theorem 1: *The capacity of a concentrator is the minimum value of* c which satisfies

$$H_c = \sum_{i=1}^{c} h_i > c ,$$

where $(h_1,h_2,\ldots,h_{|O|})$ is the isolation vector of the concentrator.

Proof: If c is the minimum value which satisfies $H_c > c$, then any H_j for $1 \leq j < c$ is less than or equal to j , that is , $H_j \leq j$ for $1 \leq j < c$. Therefore any subset I' of I , where $|I'|=j<c$, has crosspoints to a subset $O'=\bigcup_{i \in I'} u(i) \leq O$ and $|O'| \geq j$. Thus from Hall's Theorem (see Appendix), any subset $I' \subseteq I$ of size up to and including c-1 has some subset $O' \subseteq O$ where there is a one-to-one correspondence between I' and O' . Now from the requirement on the concentrator that for any $i \in I$, $u(i) \neq \phi$, it follows that for the given subset O' , every element $i \notin \sigma(O')$ is connected at least one element in O which is not in O' ; that is,

$$\text{for all } i \notin \sigma(O') \quad , \quad u(i) \cap (O - O') \neq \phi .$$

Therefore for any subset $I' \subseteq I$ up to and including size c, there is some subset of $O' \subseteq O$ which has one-to-one correspondence with I' . Hence the capacity is c .

Q.E.D.

Example 2: The isolation vector of the concentrator of Figure 1(a) is (1,2,2).

$$H_1 = h_1 = 1 \leq 1 \quad \text{while} \quad H_2 = h_1 + h_2 = 1+2 = 3 > 2 .$$

Therefore the capacity of the concentrator is 2 .

By making use of Theorem 1 it is clearly straightforward to calculate the capacity of a concentrator when the isolation vector or the values of the isolation numbers $H_j, 1 \leq j \leq |O|$, are known. But, in general, this information is not easy to obtain. However, for some special cases, such as binomial concentrators, we can give the following.

Lemma 1: The isolation vector of a $\binom{x}{y}$-concentrator is

$$\left(\binom{0}{y-1},\binom{1}{y-1},\ldots\binom{x-2}{y-1},\binom{x-1}{y-1}\right) .$$

Proof: An $\binom{x}{y}$-concentrator has $\binom{x}{y}$ input lines, while an $\binom{x-1}{y}$-concentrator (which is contained in an $\binom{x}{y}$-concentrator) has one less output line than the $\binom{x}{y}$-concentrator and only $\binom{x-1}{y}$ input lines. Therefore at most $\binom{x-1}{y}$ input lines are isolated by $x-1$ output lines. That is $H_{x-1}=\binom{x-1}{y}$. Now from the formula $\binom{a}{b}=\binom{a-1}{b}+\binom{a-1}{b-1}$, we have that

$$b_x=H_x-H_{x-1}=\binom{x}{y}-\binom{x-1}{y}=\binom{x-1}{y-1} .$$

By using this equation recursively the isolation vector is obtained in descending order.

Q.E.D.

With Lemma 1, we can prove the following.

Theorem 2 (Masson [2]): The capacity of $\binom{x}{y}$-concentrator is $y+2$.

Proof: Since in a concentrator, the number of inputs is always larger than the number of outputs, it follows that $x>y+2$.

From Lemma 1 $$H_{y+1}=\binom{y+1}{y}=y+1 ,$$

and $$H_{y+2}=\binom{y+2}{y}=\frac{y+1}{y}(y+2) ,$$

Note that $$H_{y+1}=y+1<y+2<\frac{y+1}{y}(y+2)=H_{y+2}$$

Hence, the capacity is $y+2$. Q.E.D.

Composite Concentrators

Given two one-stage sparse crosspoint networks, say, $A:(I_1,O_1,u_1)$ and $B:(I_2,O_2,u_2)$, we define the composite one-stage sparse crosspoint as $A\cdot B$: (I_1xI_2,O_1xO_2,u_{12}) or more simply (I_{12},O_{12},u_{12}) where I_{12} is the Cartesian product of I_1 and I_2 and O_{12} is the Cartesian product of O_1 and O_2, and where u_{12}: $I_1xI_2\rightarrow O_1xO_2$ such that

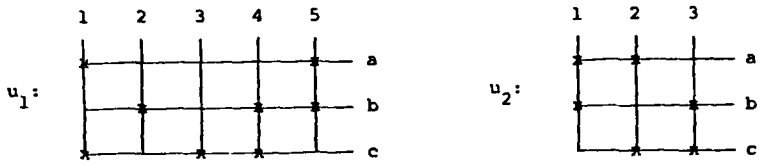
$$u_{12}(i_1i_2)=\{u_1(i_1)xu_2(i_2)\}$$

where $i_1\varepsilon I_1$ and $i_2\varepsilon I_2$ and $u_1(i_1)xu_2(i_2)$ is the Cartesian product of $u_1(i_1)$ and $u_2(i_2)$. The networks A and B will be referred to as the component networks of the composite network $A\cdot B$.

Example 3: Let A denote the network of Figure 1(a) and let B denote the network of Figure 1(c).

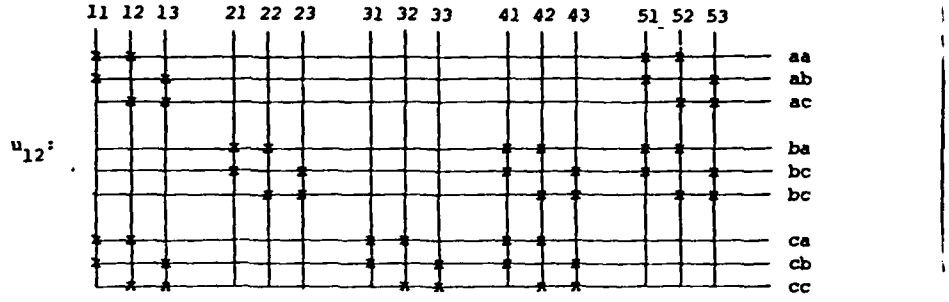
A: $(\{1,2,3,4,5\}, \{a,b,c\}, u_1)$

B: $(\{1,2,3\}, \{a,b,c\}, u_2)$



Then the composite network $A\cdot B$ has 15 inputs and 9 outputs and a composite relation u_{12}

$A\cdot B$: $(\{(1,1),(1,2),(1,3),(2,1),(2,2),(2,3),(3,1),(3,2),(3,3),(4,1),(4,2),(4,3),(5,1),(5,2),(5,3)\},\{(a,a),(a,b),(a,c),(b,a),(b,b),(b,c),(c,a)(c,b),(c,c)\},u_{12})$



Note that A•B is a composite concentrator. Hence if we knew the isolation vector of this composite concentrator, by Theorem 1 we could determine the capacity. However even though the isolation vectors of the components are known, the isolation vector of the components is not apparent. Hence, we will now define the concept of an isolation array with which to address this problem.

In general, suppose that two networks P and Q have following isolation vectors: $(p_1,p_2,\ldots,p_n)$ and $(q_1,q_2,\ldots,q_n)$. Then the isolation array for the composite network P•Q is an mxn array whose i^{th} row, j^{th} column element is equal to p_iq_j . In other words, it is the following vector product of isolation vectors:

$$\begin{bmatrix} p_1 \\ p_2 \\ \cdot \\ \cdot \\ \cdot \\ p_m \end{bmatrix} \quad (q_1,q_2,\ldots,q_n) = \begin{bmatrix} p_1q_1, & p_1q_2, & \cdots, & p_1q_n \\ p_2q_1, & p_0q_2, & \cdots, & p_2q_n \\ & & --- & \\ p_mq_1, & p_mq_2, & \cdots, & p_mq_n \end{bmatrix}$$

Example 4: The isolation array of the composite network A•B of Example 3 is

$$\begin{bmatrix} 1 \\ 2 \\ 2 \end{bmatrix} \quad (0,1,2) = \begin{bmatrix} 0 & 1 & 2 \\ 0 & 2 & 4 \\ 0 & 2 & 4 \end{bmatrix} .$$

For later reference, we will denote the i^{th} row, j^{th} column element of the isolation array for the composite network P•Q as h_{ij} . Then if P has m outputs and Q has n outputs, the isolation array is

$$\begin{bmatrix} h_{11}h_{12}h_{13}\cdots h_{1n} \\ h_{21}h_{22}h_{23}\cdots h_{2n} \\ --- \\ h_{m1}h_{m2}h_{m3}\cdots h_{mn} \end{bmatrix}$$

Capacity of Composite Concentrators

We will present in this section a theorem which is the basis of a capacity calculation technique for a class of composite concentrators. However, before doing this, we must first develop the concept of a smooth network.

We will define a network, say, P: (I,O,u) to be smooth if, given the network's isolation numbers $H_1,H_2,\ldots,H_{|O|}$, there exists a sequence of $|O|$ subsets of O, denoted as $O_1,O_2,\ldots,O_{|O|}$, where

$$O_1 \subset O_2 \subset O_3 \subset \ldots \subset O_{|O|}$$

and

$$|\sigma(O_j)| = H_j .$$

Example 5: The concentrator of Figure 1(a) (for which $H_1=1, H_2=3$, and $H_3=5$) is smooth as can be seen by examining $O_1=\{b\}$, $O_2=\{b,c\}$, and $O_3=\{a,b,c\}$ since $\sigma(\{b\})=\{2\}$, $\sigma(\{b,c\})=\{2,3,4\}$, $\sigma(\{a,b,c\}) = \{1,2,3,4,5\}$.

Finally, we will define an $m(n)$=partition of an integer c to be a partition of c into m integer parts, $r_1,r_2,\ldots,r_m$, such that $r_1+r_2+\ldots+r_m=c$, $0\leq r_j\leq n$, and $r_i\geq r_j$ for $i<j$. We can now state the followong:

Theorem 3: Given the isolation array of the composite network P•Q where the isolation vector of P is $(p_1,\ldots,p_m)$ and the isolation vector of Q is $(q_1,\ldots,q_n)$ and at least one of these two component networks is smooth, then the capacity of P•Q is given by the minimum value of c which satisfies

$$\max_{\substack{\text{over all } m(n)\text{-partitions} \\ \text{of } c}} \left(\sum_{j=1}^{r_1} h_{1_j} + \sum_{j=1}^{r_2} h_{2_j} + \ldots + \sum_{j=1}^{r_m} h_{m_j} \right) > c .$$

The proof of Theorem 3 will be omitted in this version of this paper because of its length [3].

It should be noted that the complexity of calculating the capacity of a P•Q composite concentrator by repeatedly using Theorem 3 for increasing values of c can be shown to be $O(N^{3/2})$ [5].

Example 6: For the composite concentrator A•B of Example 3, the 3(3)-partition which satisfies Theorem 3 is: 3,3,0. Hence, for the composite concentrator A•B, we have that $c=6$. It should also be noted that using Theorem 3 repeatedly by starting with $c=1$ and programming to $c=6$ results in a consideration of 15 partitions.

Capacity Observations

Using Theorem 3, a wide variety of interesting capacity observations can be made. For example, a P•Q composite concentrator where P is a $\binom{u}{v}$-concentrator and Q is a $\binom{x}{y}$-concentrator is called a $\binom{u}{v}\binom{x}{y}$-concentrator. Formulas for the capacity in terms of the parameters u,v,x, and y for all cases of interest are listed in Table 1 and shown graphically in Figures 2(a)-2(d).

Higher dimensional composite concentrators can also be considered. For example, we can determine the capacity of a $\binom{4}{2}\binom{4}{2}\binom{4}{2}$-concentrator, or more simply, a $\binom{4}{2}^3$-concentrator using a 3-dimensional isolation matrix. Similarly $\binom{4}{2}^n$-concentrators can be considered. Table 2 gives the capacity of $\binom{x}{2}^n$-concentrators for $1\leq n\leq 8$, $x\geq 4$.

Finally, Figure 3 shows a 3 dimensional isolation array for $\binom{u}{2}\binom{w}{3}\binom{y}{4}$-concentrators. Some rather interesting capacity observations for this 3 dimensional composite concentrator are the following: for $u\geq 4$, $w=4$, and $y=5$, $c=79$; for $u\geq 2$, $w=5$, and $y=5$, $c=74$; for $u\geq 2$, $u\geq 6$, and $y=5$, $c=58$; and for $u\geq 2$, $w\geq 4$, and $y\geq 6$, $c=48$.

$\binom{u}{v}\binom{x}{y}$ - Concentrator $2 \leq v \leq y$ and $\binom{u}{v}\binom{x}{y} > ux$

Case 1.	$y=2$		
Case 1.1.		$u \neq v$ and $x \neq y$	$c=(v+1)(y+2)-1$
Case 1.2.	$u=v$ or $x=y$		
Case 1.2.1.		$u=v$	$c=v(y+4)$
Case 1.2.2.		$x=y$	$c=(v+4)y$
Case 2.	$y \neq 2$		
Case 2.1.	$v=y$		
Case 2.1.1.		$u \leq v+2$	$c=(v+1)(y+2)-1$
Case 2.1.2.		$u>v+2$	$c=v(y+3)$
Case 2.2.	$v \neq y$		
Case 2.2.1.		$x=y$	$c=uy$ where u is minimum and $\binom{u-1}{v-1} > vy$
Case 2.2.2.	$x=y+1$		
Case 2.2.2.1.		$2y>v(v+1)$	$c=(v+2)(y+1)$
Case 2.2.2.2.		$2y \leq v(v+1)$	$c=(v+2)(y+1)-1$
Case 2.2.3.	$x=y+2$		
Case 2.2.3.1.		$2v>y$	$c=(v+1)(y+2)-1$
Case 2.2.3.2.		$2v \leq y$	$c=v(y+2)$
Case 2.2.4.	$x \geq y+3$		
Case 2.2.4.1.		$2v>y$	$c=v(y+3)$
Case 2.2.4.2.		$2v \leq y$	$c=v(y+2)$

Table 1

$\binom{u}{v}\binom{x}{y}$ - Concentrator $2 \leq v \leq y$ and $\binom{u}{v}\binom{x}{y} > ux$

Case 1. $y=2$

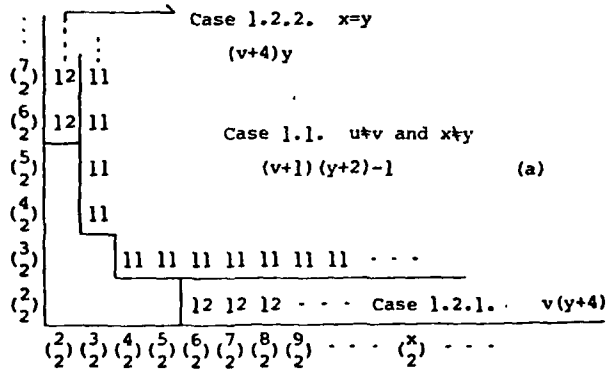
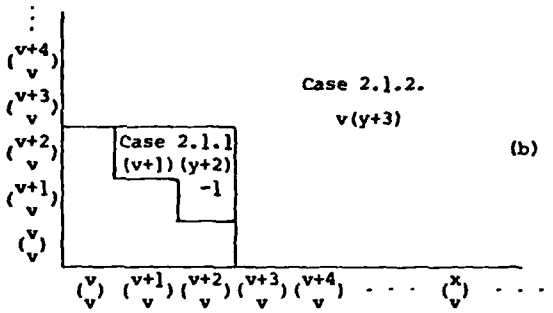


Figure 2

Case 2. $y \neq 2$

Case 2.1. $v=y$



Case 2. $y \neq 2$ (continued)

Case 2.2. $v=y$

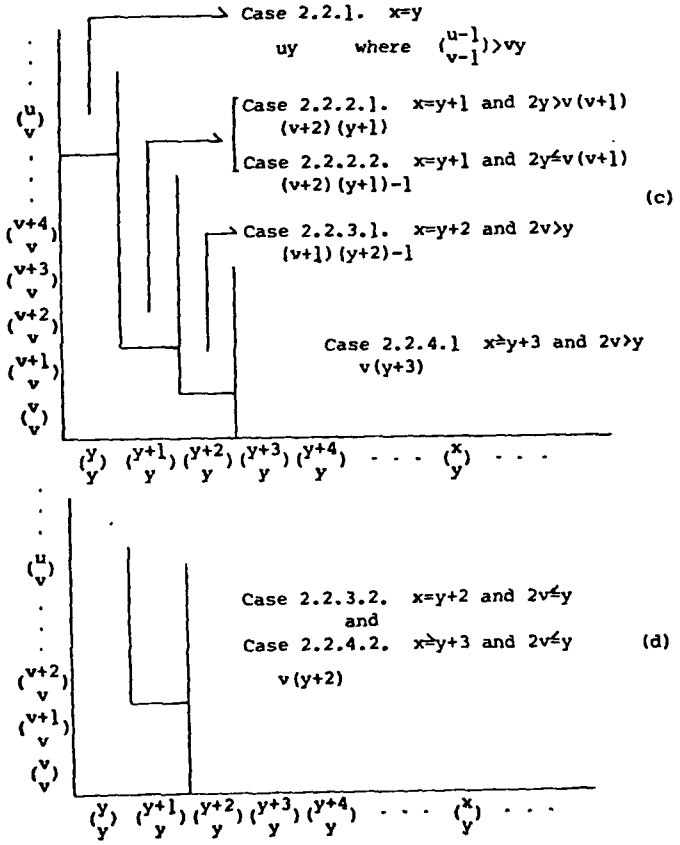


Figure 2 (continued)

n	1	2	3	4	5	6	7	8
capacity	4	11	32	96	286	826	2588	7762

Table 2

ACKNOWLEDGMENT

The concept of a composite concentrator was first described by the authors and Mr. George C. Gingher in [1]. The authors wish to acknowledge useful discussion with Mr. Gingher regarding this topic.

REFERENCES

[1] G. M. Masson, G. C. Gingher, and S. Nakamura, "A Sampler of Circuit Switching Networks, Computer, Vol. 12, No. 6, pp. 32-48, June, 1979.

[2] G. M. Masson, "Binomial Switching Networks for Concentration and Distribution," IEEE Trans. Comm., Vol. COM-25, No. 9, pp. 873-883, Sept., 1977.

[3] S. Nakamura and G. M. Masson, "Capacity Calculation for a Class of Composite Concentrators," Johns Hopkins University, Electrical Engineering Report 79-9, 1979.

[4] P. Hall, "On Representatives of Subsets," J. London Math. Soc., Vol. 10, pp. 26-30, 1935.

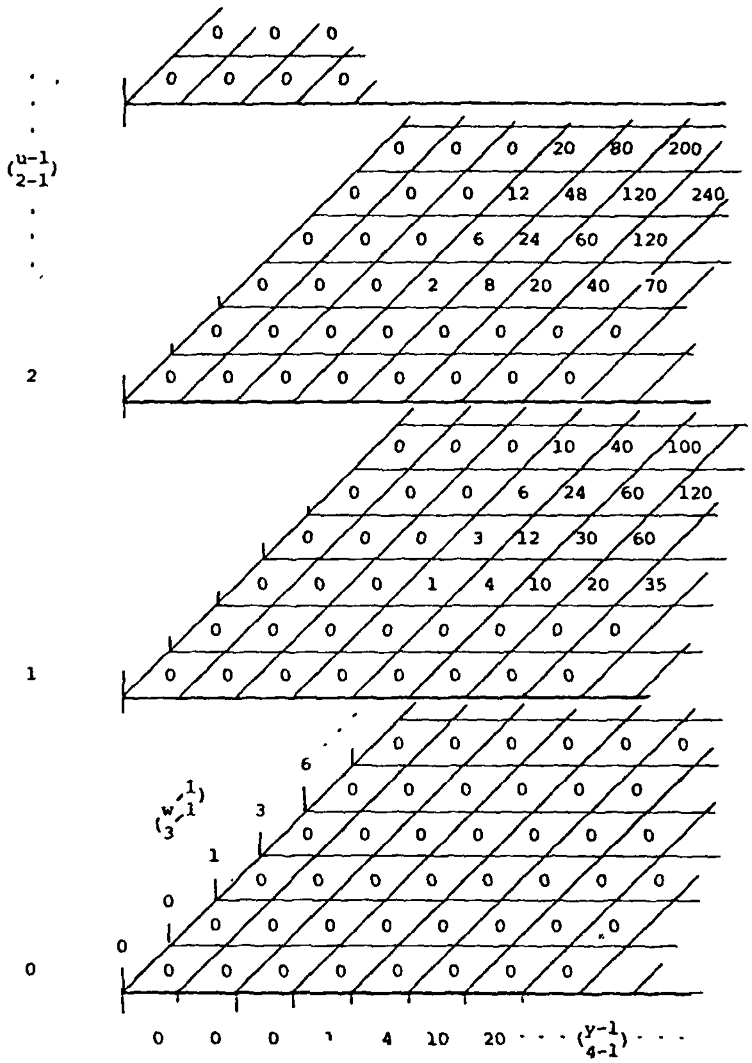
[5] E. M. Wright, Partitions of Multipartite Number into k Parts, J. Reine Angew. Math., Vol. 216, pp. 101-112, 1964.

APPENDIX

Hall's Theorem [4]:

Any set of inputs I', $I' \subseteq I$, has a one-to-one corresponsence with some output set $O' \subseteq O$ if and only if for any subset $I'' \subseteq I'$ we have that

$$|u(I'')| \geq |I''| .$$



0 0 0
0 0 0 0
$\binom{u-1}{2-1}$
0 0 0 20 80 200
0 0 0 12 48 120 240
0 0 0 6 24 60 120
0 0 0 2 8 20 40 70
0 0 0 0 0 0 0 0
2
0 0 0 0 0 0 0 0
0 0 0 10 40 100
0 0 0 6 24 60 120
0 0 0 3 12 30 60
0 0 0 1 4 10 20 35
0 0 0 0 0 0 0 0
1
0 0 0 0 0 0 0 0
0 0 0 0 0 0
6
$\binom{w-1}{3-1}$
3
0 0 0 0 0 0
1
0 0 0 0 0 0 0
0
0 0 0 0 0 0 0 0
0
0 0 0 0 0 0 0 0
0
0 0 0 0 0 0 0 0
0 0 0 1 4 10 20 . . . $\binom{y-1}{4-1}$ . . .

Figure 3

HIGHER ORDER COMPOSITE CONCENTRATORS

by

Shinji Nakamura and Gerald M. Masson
Department of Electrical Engineering
The Johns Hopkins University
Baltimore, Maryland 21218

Abstract

A concentrator is a type of interconnection network which can provide disjoint paths by means of closing switches (usually called crosspoints) from any specified set of inputs to the network (where this input set has cardinality less than or equal to some number called the capacity) to some arbitrary set of outputs of the same size. In this paper, we consider the calculation of the capacity of a class of concentrators called composite concentrators. Composite concentrators are one-stage interconnection networks with sparse crosspoint patterns between the inputs and outputs where this total crosspoint pattern is determined from the crosspoint patterns of smaller, component concentrators which have the so-called binomial crosspoint pattern.

Introduction

A concentrator is a type of interconnection network which can provide disjoint paths from any specified set of inputs to the network (where this input set has cardinality less than or equal to some number called the capacity) to some arbitrary set of outputs of the same size. The crucial limitation of this type of interconnection network is that the output set to which the specified inputs can be connected cannot be arbitrarily specified. Indeed, the assignment of a given input to a particular output is often not possible. For example Figure 1 shows a design of one-stage, six input, four output concentrator. In this design an "x" represents a crosspoint between input and

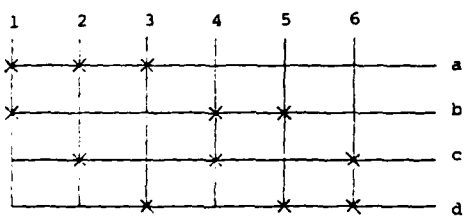


FIGURE 1

output lines indicating that there can be a connection between the lines. Note that the input set $\{1,2\}$ can be connected to any one of the output sets $\{a,b\}$, $\{a,c\}$, or $\{b,c\}$. But it is clear that there is no means of connecting $\{1,2\}$ to, say, $\{a,d\}$. Hence, there is limited access to the outputs from the inputs. Nevertheless, it will be shown later that this concentrator, called a binomial $\binom{4}{2}$-concentrator (since the network's column crosspoint pattern displays exactly once every possible combination of placing two crosspoints on the four rows), can connect any four (or fewer) inputs to some output set of the same size. Hence, this network has a capacity of 4.

Sparse crosspoint placement in a one-stage network to provide a maximum capacity is the main topic of this paper. More particularly, we will consider an extension of the binomial design method for concentrators. The binomial $\binom{u}{v}$-concentrator and its associated capacity were first discussed in [1] and [2]. An extension of this design, the composite binomial concentrator, was discussed in [2] and [3]. In this paper, a further extension of this design, called a higher order binomial composite concentrator, is considered.

Preliminaries

A one-stage concentrator can be defined as a triplet (I,O,r), where I is the input set, O is the output set, and r is the crosspoint placement relation between I and O. For the example of Figure 1, $I = \{1,2,3,4,5,6\}$, $O = \{a,b,c,d\}$, and $r = \{(1,a), (1,b), (2,a), (2,c), (3,a), (3,d), (4,b), (4,c), (5,b), (5,d), (6,c), (6,d)\}$. r can be expressed by a graph as well as by a crosspoint diagram as shown in Figure 1. When only a convenient summary of the properties of a concentrator need be expressed, another triplet of numbers $(|I|,|O|,c)$ where $|I|$ is the number of inputs, $|O|$ is the number of outputs, and c is the capacity of the concentrator can be used. Recall that the network has capacity c if any k inputs, $k \leq c$, can be connected to some k outputs using k of the crosspoints. For the concentrator of Figure 1, this triplet is $(6,4,4)$.

A binomial $\binom{u}{v}$-concentrator is a one-stage concentrator having $\binom{u}{v}$ inputs, u outputs, and where each of the $\binom{u}{v}$ inputs has r crosspoints to a unique choice of v of the u outputs.

A composite concentrator $C_1 \times C_2 = (I_1 \times I_2, O_1 \times O_2, r_{12})$ of two component networks $C_1 = (I_1, O_1, r_1)$ and $C_2 = (I_2, O_2, r_2)$ is a one-stage concentrator where $I_1 \times I_2$ and $O_1 \times O_2$ are Cartesian products, and r_{12} is the crosspoint placement relation between $I_1 \times I_2$ and $O_1 \times O_2$ which is defined as

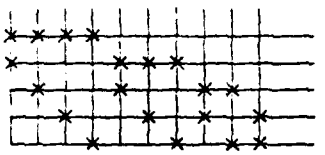
$$r_{12}(i_1 i_2) = \{r_1(i_1) \times r_2(i_2)\},$$

for $i_1 \in I_1$ and $i_2 \in I_2$. Figure 2 shows a $\binom{5}{2}\binom{3}{2}$-composite concentrator which was composed from a $\binom{5}{2}$-concentrator and a $\binom{3}{2}$-concentrator. Higher order concentrators, $C_1 \times C_2 \times \cdots \times C_n$, are defined in a similar manner.

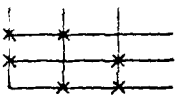
Isolation Arrays

For an input $i \in I$ of a single concentrator (I,O,r), $r(i)$ is the set of outputs to which i has crosspoints and is therefore capable of being connected. Hence, without all the outputs of $r(i)$, the input i is isolated from the outputs. Looking at this from the output side, for any subset O' of the outputs O, the issue is what set of inputs are isolated by O'. We will express such a subset of inputs as $\sigma(O')$. Hence $\sigma(O')$ is

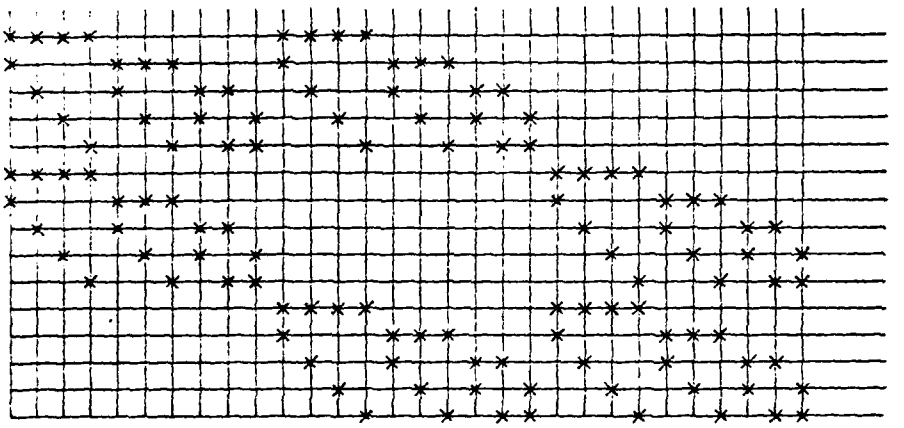
$$\sigma(O') = \{i \mid i \in I \text{ and } r(i) \subseteq O'\}.$$



$\binom{5}{2}$-concentrator



$\binom{3}{2}$-concentrator



$\binom{5}{2}\binom{3}{2}$-composite concentrator

FIGURE 2

Now for all subsets of outputs of size k, we define the isolation number H_k as:

$$H_k = \max_{O' \subseteq O \text{ and } |O'| = k} |\sigma(O')| .$$

That is, the isolation number is the maximum size of all the isolation sets which are isolated by any output set of size k. With isolation numbers, we can define an isolation vector as:

$$S = (h_1, h_2, \ldots, h_{|O|})$$

where $h_j = H_j - H_{j-1}$ for $1 \leq j \leq |O|$ and H_0 is zero.

A higher order isolation array for composite concentrators is defined by the array multiplication of isolation vectors. To begin, the two dimensional isolation array of two isolation vectors, say, S_1 and S_2 is obtained from ordinary vector multiplication. More generally, the elements of a m^{th} order isolation array of m vectors $S_1 \times S_2 \times \ldots \times S_m$ are defined by products such as:

$$h_{j_1 j_2 \ldots j_m} = h^1_{j_1} h^2_{j_2} \ldots h^i_{j_i} \ldots h^m_{j_m} ,$$

where $h^i_{j_i}$ is the j_i^{th} element of the S_i isolation vector.

Capacity Calculation

The capacity of a concentrator can be calculated by means of its isolation vector.

Theorem 1 [3]: The capacity of a concentrator is the minimum value of c which satisfies

$$H_c = \sum_{i=1}^{c} h_i > c ,$$

where $(h_1, h_2, \ldots, h_{|O|})$ is the isolation vector of the concentrator.

For a binomial $\binom{u}{v}$-concentrator, the isolation vector can be shown to be [3]:

$$\left(\binom{0}{v-1}, \binom{1}{v-1}, \ldots, \binom{u-2}{v-1}, \binom{u-1}{v-1}\right) .$$

This leads simply to:

Theorem 2 [2], [3]: The capacity of $\binom{u}{v}$-concentrator is $v+2$.

This can be extended to second order, binomial, $\binom{u}{v}\binom{w}{x}$-concentrators as follows:

Theorem 3: The capacity of a binomial $\binom{u}{v}\binom{w}{x}$-concentrator is the minimum value of c which satisfies

$$\sum^{c} h_{ij} > c ,$$

where the summation is over c of the h_{ij}'s, where if h_{ij} is in the summation, then all $h_{k\ell}$'s, $k \leq i$ and $\ell \leq j$, are also in the summation.

For the special class of a second order binomial $\binom{v+2}{v}\binom{v+2}{v}$-concentrator or, more simply, a $\binom{v+2}{v}^2$-concentrator the isolation array is the two dimensional array shown in Figure 3. Now, the summation of h_{ij}'s up to and including $h_{(v+1),(v+1)}$ satisfying the ordering required by Theorem 3 yields a sum which is equal to the number of elements added. That is,

Theorem 5: The capacity c of a n^{th} order $(\frac{u}{v})^n$-concentrator is bounded as follows:

$(v+1)^n < c \leq v\cdot(v+1)^{n-3}(v+2)^2$ for $u=v+2$ and $n \geq 3$,

$(v+1)^n < c \leq v\cdot(v+1)^{n-2}(v+3)$ for $u \geq v+3$ and $n \geq 2$.

Proof: Part 1: $(v+1)^n < C$

For the two dimensional isolation array case, the maximum sum of $(v+1)^2$ elements satisfying the ordering required by Theorem 4 can be shown to be

$$\sum_{i=1}^{v+1}\sum_{j=1}^{v+1} h_{ij} = (v+1)^2 .$$

This generalizes to the n^{th} order isolation array case where it can be shown that the maximum sum of $(v+1)^n$ elements is

$$\sum_{j_1=1}^{v+1}\sum_{j_2=1}^{v+1}\cdots\sum_{j_n=1}^{v+1} h_{j_1 j_2 \cdots j_n} = (v+1)^n .$$

Now, this part of the proof will be completed if it can be shown that there is no way of choosing $m < (v+1)^n$ elements which sum to more than m . To show this, it will first be convenient to simplify the notation. We will therefore write the isolation vector

$$(\binom{0}{v-1},\ldots,\binom{v-2}{v-1},\binom{v-1}{v-1},\binom{v}{v-1},\binom{v+1}{v-1},\binom{v+2}{v-1},\ldots,\binom{v+n-2}{v-1},\binom{v+n-1}{v-1},\ldots) .$$

more simply as

$$(0,\ldots,0,1,v,h_2,h_3,\ldots,h_{n-1},h_n\ldots) .$$

Using this, the two dimensional isolation array can be given as shown in Figure 5.

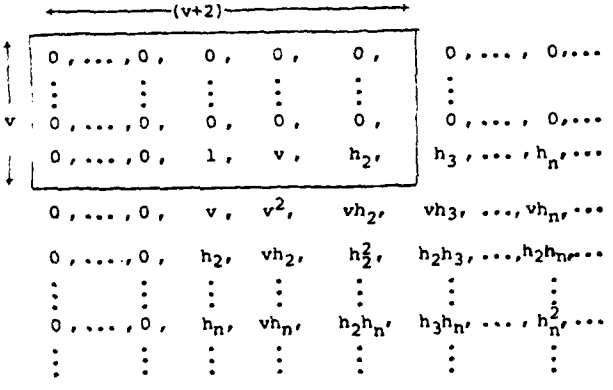


Figure 5

Clearly (and as indicated in Figure 5), the minimum number of elements that must be summed to include h_2 in the summation is

$$v\cdot(v+2) = v^2 + 2v < (v+1)^2 .$$

Moreover, this is the only possible selection of less than $(v+1)^2$ elements that sums up to more than $(v+1)$. Hence, so as not to exceed $v\cdot(v+2)$, h_2 must be such that

$$1 + v + h_2 \leq v(v+2) = v^2 + 2v .$$

Similarly, for higher order cases,

$$1 + v + h_2 + h_3 + \ldots + h_i \leq v^i + iv^{i-1} .$$

Noting that

$$1 + v + h_2 + h_3 + \ldots + h_i = \binom{v+i}{v} ,$$

and that

$$\binom{v+i}{v} = \frac{(v+1)(v+2)\ldots(v+i)}{i!} < v^i + iv^{i-1} ,$$

We can conclude that there are not $m < (v+1)^n$ elements of the isolation array that will sum to more than m . Hence $(v+1)^n < c$.

Part 2:

$$c \leq v\cdot(v+1)^{n-3}(v+2)^2 \quad for \quad u=v+2 \quad and \quad n \geq 3 \quad (a)$$

and

$$c \leq v\cdot(v+1)^{n-2}(v+3) \quad for \quad u \geq v+3 \quad and \quad n \geq 2 \quad (b)$$

These two inequalities can be proven by showing that there always exists a set of m elements which sums to more than m when m is equal to $v\cdot(v+1)^{n-3}(v+2)^2$ or $v\cdot(v+1)^{n-2}(v+3)$ respectively. Considering (b) of Part 2 first, we have previously shown that in the two-dimensional case that there exists $v\cdot(v+3)$ elements satisfying the subscript ordering requirement that sum to more than $v\cdot(v+3)$. Now, to go to higher dimensional isolation arrays, consider the first $v+1$ elements of the isolation vector

$$(0 , 0 , \ldots , 0 , 1 , v , \ldots) .$$
$$\longleftarrow \quad v+1 \quad \longrightarrow$$

These first $v+1$ elements clearly sum to $v+1$ so as we expand to higher dimensional arrays $(h \geq 3)$, it is clear that we can choose for each additional dimention $(v+1)$ times the number of elements of the previous dimension which sum to $(v+1)$ times the previous sum. In other words, for the n^{th} order case, if we select $v\cdot(v+1)^{n-2}(v+3)$ elements, $h_{j_1 j_2 \cdots j_n}$, where $1 \leq j_1 \leq v$, $1 \leq j_2 \leq (v+3)$, and $1 \leq j_k \leq (v+1)$, $3 \leq k \leq n$, it follows that they will sum to more than $v\cdot(v+1)^{n-2}(v+3)$.

Considering next (a) of Part 2, it is easily seen that for the three dimensional case, the elements $h_{j_1 j_2 j_3}$, $1 \leq j_1 \leq v$, $1 \leq j_k \leq (v+2)$, $k = 2,3$, sum to more than $v(v+2)^2$. Then, similar to the argument for (b), expanding to higher dimensions will maintain this unequality since the number of elements and the sum increase by a factor of $(v+1)$ for each new dimension. In other words, for the n^{th} order case, if we select $v\cdot(v+1)^{n-3}(v+2)^2$ elements, $h_{j_1 j_2 \cdots j_n}$, where $1 \leq j_1 \leq v$, $1 \leq j_k \leq (v+2)$, $k = 2,3$, and $1 \leq j_\ell \leq v+1$, $4 \leq \ell \leq n$, it follows that they will sum to more than $v\cdot(v+1)^{n-3}(v+2)^2$.

Q.E.D.

Beyond the capacity bounds of Theorem 5 for $(\frac{u}{v})^n$-concentrators, an essentially similar argument to that of Theorem 4 yields the exact capacity for sufficiently large n . This result is stated without proof as

Theorem 6: The capacity of $(\frac{u}{v})^n$-concentrator for sufficiently large n is

$$v(v+1)^{n-3}(v+2)^2 - \binom{n-2}{n}+4 \quad for \quad u = v+2$$

or

$$v(v+1)^{n-2}(v+3) - v(n+5) \quad for \quad u \geq v+3 .$$

$$\begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ v \\ \frac{1}{2}v(v+1) \end{pmatrix} (0,\ldots,0,1,v,\tfrac{1}{2}v(v+1)) = \begin{bmatrix} 0\ldots0 & 0 & 0 & 0 \\ \vdots\ \ \vdots & \vdots & \vdots & \vdots \\ 0\ldots0 & 1 & v & \frac{v(v+1)}{2} \\ 0\ldots0 & v & v^2 & \frac{v^2(v+1)}{2} \\ 0\ldots0 & \frac{v(v+1)}{2} & \frac{v^2(v+1)}{2} & \frac{v^2(v+1)^2}{4} \end{bmatrix}$$

Figure 3

$$\begin{bmatrix} 0\ldots0 & 0 & 0 & 0 & 0 & 0 \\ \vdots\ \ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0\ldots0 & 0 & 0 & 0 & 0 & 0 \\ 0\ldots0 & 0 & 1 & v & \frac{v(v+1)}{2} & \frac{v(v+1)(v+2)}{6} \\ 0\ldots0 & 0 & v & v^2 & \frac{v^2(v+1)}{2} & \frac{v^2(v+1)(v+2)}{6} \\ 0\ldots0 & 0 & \frac{v(v+1)}{2} & \frac{v^2(v+1)}{2} & \frac{v^2(v+1)^2}{4} & \frac{v^2(v+1)^2(v+2)}{12} \\ 0\ldots0 & 0 & \frac{v(v+1)(v+2)}{6} & \frac{v^2(v+1)(v+2)}{6} & \frac{v^2(v+1)^2(v+2)}{12} & \frac{v^2(v+1)^2(v+2)^2}{36} \end{bmatrix}$$

Figure 4

$$\sum_{i=1}^{v+1}\sum_{j=1}^{v+1} h_{ij} = 1+v+v+v^2 = (v+1)^2 ,$$

and this is the maximum value of any summation of $(v+1)^2$ terms in the allowable adding order. The next nonzero element, $h_{v,(v+2)}$, is v elements away with a separation of $v-1$ zeros. But

$$h_{v,(v+2)} = \frac{v(v+1)}{2} > v$$

Thus,

$$\sum_{i=1}^{v+1}\sum_{j=1}^{v+1} h_{ij} = (v+1)^2 < (v+1)^2 + v = (v+1)(v+2)-1 < (v+1)^2 + \frac{v(v+1)}{2} .$$

Therefore from Theorem 3, the capacity of the $\binom{v+2}{v}^2$-concentrator is $(v+1)(v+2)-1$. Similarly, the isolation array for the $\binom{v+3}{v}^2$-concentrator is shown in Figure 4. The maximum summation of $v\cdot(v+3)$ elements of the isolation array satisfying the required ordering is

$$\sum_{i=1}^{v}\sum_{j=1}^{v+3} h_{ij} = 1+v+\frac{v(v+1)}{2}+\frac{v(v+1)(v+2)}{6} > v(v+3) .$$

While for $v \geq 3$

$$v(v+3) > \frac{1}{2}v(v+1)+1 = \sum_{i=1}^{v}\sum_{i=1}^{v+3} h_{ij} - h_{v,(v+3)} .$$

Therefore from Theorem 3, for $v \geq 3$, the capacity of $\binom{v+3}{v}^2$-concentrator is $v(v+3)$.

Since it is easily seen that any $\binom{u}{v}^2$-concentrator with $u \geq v+3$ has an isolation array in which the isolation array of $\binom{v+3}{v}^2$-concentrator is embedded, it can be concluded that the capacity of any $\binom{u}{v}^2$-concentrator with $v \geq 3$ and $u \geq v+3$ is $v(v+3)$.

Extensions to Higher Orders

Theorem 3 can be extended to the n^{th} order binomial concentrator case as follows.

Theorem 4: The capacity of an n^{th} order binomial $\binom{u_1}{v_1}\binom{u_2}{v_2}\ldots\binom{u_n}{v_n}$-concentrator is the minimum value of c which satisfies

$$\sum^{c} h_{j_1 j_2 \ldots j_n} > c ,$$

where the summation is over c of the $h_{j_1 j_2 \ldots j_n}$'s where if $h_{k_1 k_2 \ldots k_n}$ is in the summation, then all $h_{\ell_1 \ell_2 \ldots \ell_n}$'s, $1 \leq i \leq n$ $k_i \geq \ell_i$, are also in the summation.

In general the number of summations of h's which satisfy the conditions of Theorem 4 is proportional to an exponential function of c. Therefore, exhaustively examining all of them to find the minimal is impractical. Fortunately, for binomial $\binom{u}{v}^n$-concentrators, capacity formulas for sufficiently large n can be obtained.

Third Order Concentrators

The capacity of the general class of $\binom{u}{v}\binom{w}{x}\binom{y}{z}$-concentrators is worth considering as it demonstrates how the capacities of composite concentrators vary according to the parameters of the component concentrators (which in this case are u,v,w,x,y, and z). Accordingly, Figure 5 gives the capacity of some $\binom{u}{2}\binom{w}{3}\binom{y}{4}$-concentrators. Each of these capacities was calculated with a program that examines $O(v \times z^2)$ summations of elements of the three dimensional isolation array (all of which satisfy, of course, the subscript ordering requirement of Theorem 4).

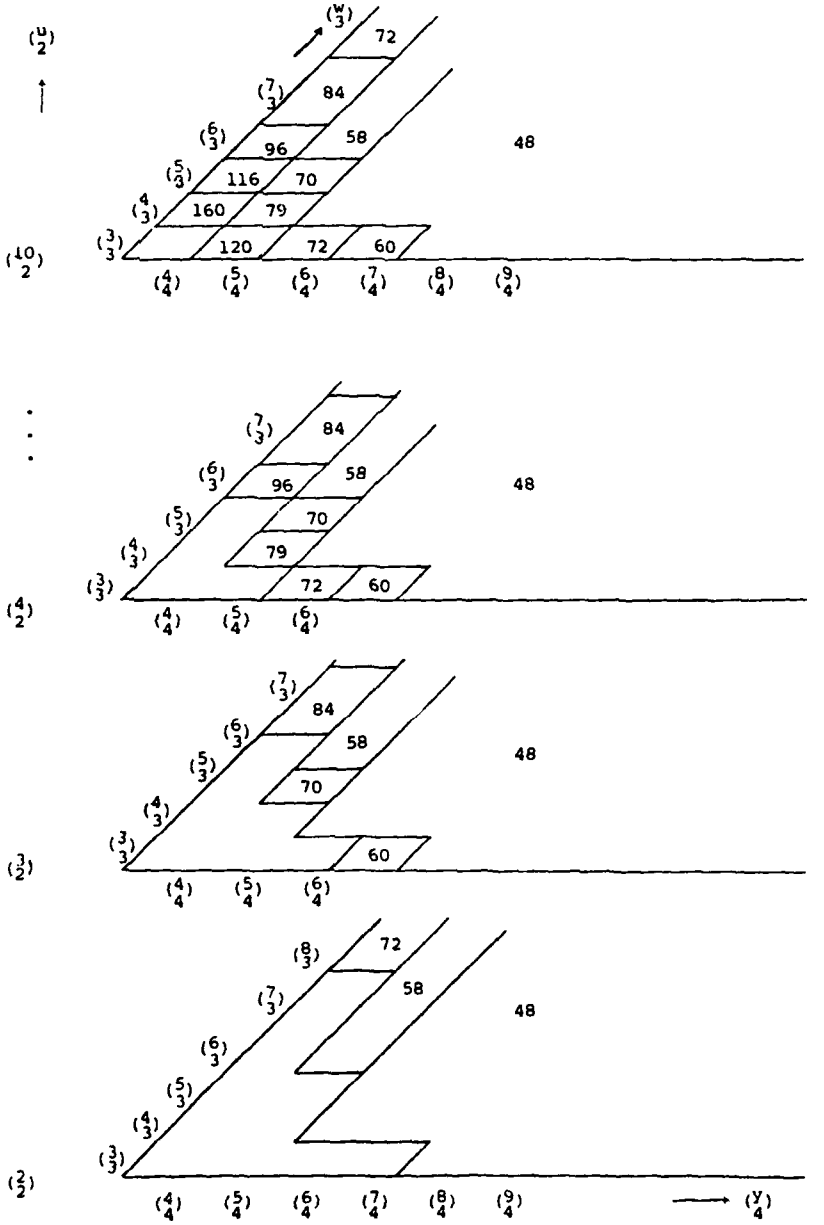


FIGURE 6

References

[1] G. M. Masson, G. C. Gingher, and S. Nakamura, "A Sampler of Circuit Switching Networks, Computer, Vol. 12, No. 6, pp. 32-48, June, 1979.

[2] G. M. Masson, "Binomial Switching Networks for Concentration and Distribution," IEEE Trans. Comm., Vol. COM-25, No. 9, pp. 873-883, September, 1977.

[3] S. Nakamura and G. M. Masson, "Capacity Calculation of Composite Concentrators," Proceedings of the Seventeenth Annual Allerton Conference on Communication, Control, and Computing, pp. 1016-1025, October, 1979.

Proceedings of the 1980 IEEE International Conference on Circuits and Computers.

ON THE DESIGN OF CONCENTRATOR NETWORKS

Gerald M. Masson and Shinji Nakamura
Department of Electrical Engineering
The Johns Hopkins University
Baltimore, Maryland 21218

ABSTRACT

An (N,M,C)-concentrator is an interconnection network with N inputs, M outputs ($M \leq N$) where for any specified subset of $C \leq M$ inputs, each of the inputs can simultaneously be connected by means of a disjoint path to some output. In this paper we consider concentrator design from the point of view of (i) lower bounds on the number of switching elements called *crosspoints* needed to implement a one-stage concentrator, and (ii) the placement of these crosspoints on a crossbar grid to produce sparse crossbar networks which function as concentrators. More particularly, we show minimal crosspoint designs for the special full capacity cases where $M = C$. With a new lower crosspoint bound that we derive for general N, M, and C, we show that the so-called binomial network is a minimal concentrator design, and we consider the near-minimality of composite binomial designs. Finally, we use our derived lower bound to show that Pippenger's construction for a N input-N output superconcentrator, which recursively uses concentrators as network components, requires at least 17.0625N crosspoints.

INTRODUCTION

An *(N,M,C)-concentrator network* is an interconnection network with N inputs and M outputs ($M \leq N$) where for any specified subset of $C \leq M$ inputs, each of the inputs can simultaneously be connected by means of a disjoint path to some output. C is referred to as the capacity of the concentrator [1]-[6]. The crucial limitation of the connecting capability of this type of interconnection network is that given a specified set of inputs, in order to simultaneously connect each input in this set of an output, the outputs cannot in general be arbitrarily specified (as would be the case in a permutation network).

Concentrators are becoming increasingly important in distributed computing systems requiring the establishment of disjoint communication paths between a subset of a large set of not necessarily identical devices and a smaller set of identical devices. For example, suppose that each input was connected to a terminal device and each output was connected to a computing device. At any one time, up to C users could each be at some subset of the N terminal devices, each requesting a connection to a computing device. Clearly, since all the computing devices are identical, it would not matter which computing device was connected to each terminal device. Concentrators satisfy this type of interconnection requirement.

The actual connections of inputs to outputs in a concentrator are accomplished by means of a switch mechanism referred to as a crosspoint. The detailed implementation of an actual crosspoint is, of course, application dependent. In the following a crosspoint can simply be considered to be a switching element when "closed" provides a connecting path between the line entering it and the line leaving it, as opposed to when it is "open" and no such connecting path exists between these two lines.

A concentrator is called a one-stage concentrator if it is realized in the form of a so-called crossbar network where between any input and any output there is at most one crosspoint. In the following it should be understood that we will be concerned only with one-stage concentrators. Obviously, a complete *crossbar network* in which there is a crosspoint between each of the N inputs and M outputs can operate as a (N,M,M)-concentrator. However, since NM crosspoints is usually a prohibitively high number, we will consider sparse crossbar networks in which there is at least one input and output between which there is no crosspoint. Clearly, the number of crosspoints and their placement are the crucial issues relative to the capabilities of sparse crossbar networks. Accordingly, our concern in the following will be with the total number of crosspoints and specifications of the outputs to which each input has crosspoints or, equivalently, specifications of the inputs to which each output has crosspoints on a crossbar grid such that the resulting network functions as a (N,M,C)-concentrator. Such specifications will be referred to as a design of that concentrator.

Preliminaries

In order to consider sparse crossbar networks as concentrator designs, we will need the following notation and definitions. Any one-stage crossbar network can be described as a triplet (I,O,R), where I is the input set, $|I| = N$, O is the output set, $|O| = M$, and R is a relation between I and O where for $i \in I$ and $o \in O$, $o \in R(i)$ implies there exists a crosspoint between input i and output o. This will at times be expressed in the following by the statement that input i is incident to output o, or, equivalently, that output o is incident to input i. R can graphically be expressed as a crosspoint diagram such as that shown in Figure 1. In this figure, (which will later be identified as a $(\frac{6}{2})$-network) $I = \{1,2,3,4,5,6\}$, $O = \{a,b,c,d\}$, and $R(1) = \{a,b\}$, $R(2) = \{a,c\}$, $R(3) = \{a,d\}$, $R(4) = \{b,c\}$, $R(5) = \{b,d\}$, $R(6) = \{c,d\}$. An "x" on the grid between input i and output o implies that $o \in R(i)$.

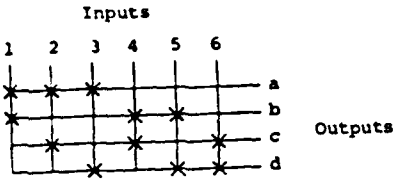


Fig. 1. A $(\binom{4}{2})$-network.

A particular network design wil ...erred to often in the following. If $I =$ ... $N\}$, and $\bigcup_{i=1}^{N} R(i)$ is the set of all poss... es of *of size* v of the elements of $O = \{$ ... and $R(i) \neq R(j)$ for all $i \neq j$, then the ...ing sparse crossbar network is called a ... t- ...work or, in this case, a $(\binom{M}{v})$-network. ... is a $(\binom{4}{2})$-network. It can be shown [3] th... $\binom{4}{2}$)- network is a design of a (6,4,4)-conce...

A subnetwork of the network describ... $\{I,O,R\}$ is simply a network described b... (I',O',R') where $I' \subseteq I$, $O' \subseteq O$, $R' \subseteq R$.

The isolation set, $\sigma(O')$, of a subset $O' \subseteq O$ of a network described by (I,O,R) is the subset $I' \subseteq I$ consisting of inputs incident to outputs only in O'. That is,

$$\sigma(O') = \{i \mid i \in I \text{ and } R(i) \subseteq O'\}.$$

Full Capacity Concentrators

An (N,M,C)-concentrator where $C = M$ is called a full capacity concentrator. The following is a necessary condition on designs of full capacity concentrators.

Theorem 1: *The minimal number of crosspoints required in the design of an (N,M,M)-concentrator is $(N-M+1)M$.*

Proof: Assume that some output, say, $o \in O$ of a design of an (N,M,M)-concentrator was incident to $N-M$ inputs. This means there are $N-(N-M) = M$ inputs with no crosspoints to o. But, clearly this contradicts the assumption that this design functions as a (N,M,M)-concentrator, since these M inputs only have crosspoints, and, therefore, can be connected to, at most $M-1$ outputs. Hence, in a (N,M,M)-concentrator design, each output must be incident to at least $N-M+1$ inputs. This results in a total of at least $(N-M+1)M$ crosspoints in any design. Q.E.D.

Designs of (N,M,M)-concentrators having exactly this minimal required number of crosspoints can be given. Clearly, in such designs, the placement of these $(N-M+1)M$ crosspoints must be such that for any choice of M of the N inputs, all subsets of size $M' \leq M$ of these chosen M inputs must collectively be incident to a total of at least M' outputs. Figures 2(a) and 2(b) give two designs of a (6,4,4)-concentrator satisfying this condition and having the minimal number of 12 crosspoints. These two designs can obviously be generalized for any (N,M,M)-concentrators. It should be noted that these designs are non-isomorphic in the sense that one cannot be transformed into the other by row/column permutations.

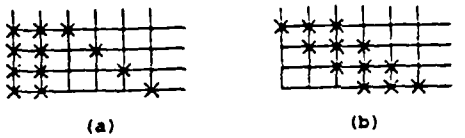


Fig. 2. Two designs of a (6,4,4)-concentrator.

In addition to the two designs of Figure 2, for certain values of N and M, other non-isomorphic designs of full capacity concentrators can be given. We have already stated that the binomial $(\binom{4}{2})$-network of Figure 1 was a design of a (6,4,4) concentrator [3]. Note that this design has the minimal number of 12 crosspoints. More generally, for all $v \geq 2$, $(\binom{v+2}{v})$-networks are designs of $((\binom{v+2}{v}), v+2, v+2)$-concentrators [3]. It is clear that these designs have the minimal number of crosspoints since

$$((\binom{v+2}{v})-(v+2)+1)\cdot(v+2) = v\cdot(\binom{v+2}{v}).$$

In some cases a subnetwork of a $(\binom{v+3}{v})$-network can be determined which is a design of a minimal full capacity concentrator. It can be shown that a $(\binom{v+3}{v})$-network is a design of a $((\binom{v+3}{v}), v+3, v+2)$-concentrator [3]. For certain choices of v, by deleting inputs from such a network, minimal designs for full capacity concentrators with $v+3$ outputs can result. For example, Figure 3(a) shows a $(\binom{6}{3})$-network which is a design of a (20,6,5)-concentrator. Now, by deleting the inputs 1,2,6,9, 10,13,14,16,17, and 18 (where the input numbering is from left to right), the network of Figure 3(b)

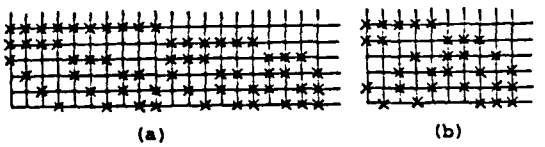


Fig. 3 (a). A $(\binom{6}{3})$-network which is a design for a (20,6,5)-concentrator;
(b). a design for a (10,6,6)-concentrator.

results. It can be shown that this resulting network is a design of a (10,6,6)-network. Moreover, it can be shown that there are at least 144 ways to delete 10 inputs of the $(\binom{6}{3})$-network to produce non-isomorphic minimal designs of a (10,6,6)-concentrator.

Finally, the number of non-isomorphic minimal designs of a full capacity (N,M,M)-concentrator for any N and M is an open question. Clearly, we could always find at least two. We will see in the following sections that for the more general (N,M,C)-concentrators, $M > C$, this is not necessarily the case.

A Lower Bound for (N,M,C)-Concentrators

In the previous section, we showed minimal designs of (N,M,C)-concentrators for the special case where $M = C$. In this section we will give a new lower bound on the number of crosspoints required in any (N,M,C)-concentrator, $M \geq C$. It will be seen that in some cases designs satisfying this lower bound can be given.

Consider any design of a (N,M,C)-concentrator. Assume that this design contains Nx crosspoints, where x is the average number of crosspoints for input. That is, on the average, each input is incident to x outputs. In the sparse crossbar network corresponding to this design, it should be clear that, since the network by assumption has capacity C, any choice of C of the M outputs, say, O' defines a subnetwork relative to the inputs in the isolation set, $\sigma(O')$, which functions as a full capacity concentrator. Let $\{O_1, O_2, \ldots, O_{\binom{M}{C}}\}$ be the $\binom{M}{C}$ possible choices of output sets of size C. Consider $\sigma(O_i)$, $i=1,\ldots,\binom{M}{C}$, and let S_i be the sum of the number of outputs incident to each of the inputs in $\sigma(O_i)$. Since the inputs of $\sigma(O_i)$ are by definition isolated by O_i, we can write that

$$S_i = \sum_{j \in \sigma(O_i)} |R(j)| .$$

It should be clear that S_i is the total number of crosspoints between the inputs of $\sigma(O_i)$ and the outputs of O_i. As the original network was a design for an (N,M,C)-concentrator, it follows that each of these subnetworks must each be a design for a $(|\sigma(O_i)|,C,C)$-concentrator $i=1,\ldots,\binom{M}{C}$. So, by Theorem 1, for all $i=1,\ldots,\binom{M}{C}$,

$$(|\sigma(O_i)|-C+1)\cdot C \leq S_i . \tag{1}$$

Summing over all $\binom{M}{C}$ choices of output sets of size C, and dividing by $\binom{M}{C}$ yields

$$\frac{\sum_{i=1}^{\binom{M}{C}} (|\sigma(O_i)|-C+1)C}{\binom{M}{C}} \leq \frac{\sum_{i=1}^{\binom{M}{C}} S_i}{\binom{M}{C}} . \tag{2}$$

It should be noted here that (2) is a necessary condition for (1). That is, if (2) is not satisfied, then there must be at least one O_i such that (1) does not hold and, therefore, the original network could not provide a capacity of C. Now, (2) can be rewritten as

$$\left(\frac{\sum_{i=1}^{\binom{M}{C}} |\sigma(O_i)|}{\binom{M}{C}} - C+1\right)C \leq \frac{\sum_{i=1}^{\binom{M}{C}} S_i}{\binom{M}{C}} . \tag{3}$$

Consider now the following term from the left-hand-side of (3):

$$\frac{\sum_{i=1}^{\binom{M}{C}} |\sigma(O_i)|}{\binom{M}{C}} . \tag{4}$$

Clearly, this term is the average number of inputs isolated by an output set of size C in this (N,M,C)-concentrator design. An equivalent expression to (3) can be obtained as follows. Recall that each input is incident on the average to x outputs in the design. Hence, the probability that an input is isolated by C of the M outputs is

$$\frac{C}{M} \cdot \frac{C-1}{M-1} \cdot \ldots \cdot \frac{C-x+1}{M-x+1} . \tag{5}$$

Since there are N inputs, it follows that the average number of inputs isolated by an output set of size C is

$$\left(\frac{C}{M} \cdot \frac{C-1}{M-1} \cdot \ldots \cdot \frac{C-x+1}{M-x+1}\right) N . \tag{6}$$

Hence,

$$\frac{\sum_{i=1}^{\binom{M}{C}} |\sigma(O_i)|}{\binom{M}{C}} = \left(\frac{C}{M} \cdot \frac{C-1}{M-1} \cdot \ldots \cdot \frac{C-x+1}{M-x+1}\right)N . \tag{7}$$

Consider next the right-hand-side of (3). The term

$$\frac{\sum_{i=1}^{\binom{M}{C}} S_i}{\binom{M}{C}} \tag{8}$$

is the average number of crosspoints in the design between a set of inputs isolated by C outputs and those outputs. Using (6) and, again, that there are on the average x crosspoints per input in the design, it follows that

$$\frac{\sum_{i=1}^{\binom{M}{C}} S_i}{\binom{M}{C}} = \left(\frac{C}{M} \cdot \frac{C-1}{M-1} \cdot \ldots \cdot \frac{C-x+1}{M-x+1}\right) \cdot Nx . \tag{9}$$

Using (7) and (9) in (2), rearranging, and combining terms allows us to state the following:

Theorem 2: A lower bound on the number of crosspoints in any design of a (N,M,C)-concentrator is Nx where x satisfies

$$\left(\frac{C}{M} \cdot \frac{C-1}{M-1} \cdot \ldots \cdot \frac{C-x+1}{M-x+1}\right) \cdot N \cdot (C-x)+C-C^2 = 0 . \tag{10}$$

Observe by Theorem 2 that, as should be expected for the full capacity concentration case where C = M, (10) is satisfied by

$$x = \frac{(N-M+1)\cdot M}{N}$$

which agrees with Theorem 1.

Observe next that for the general binomial design of a $\binom{u}{v}$-network for a $(\binom{u}{v},u,v+2)$-concentrator where $N = \binom{u}{v}$, M = u, and C = v+2, (10) is satisfied by x = v. Hence, we have the new result that binomial networks are *minimal designs*. This is stated formally as

Corollary 1: $\binom{u}{v}$-networks are minimal designs of $(\binom{u}{v}, u, v+2)$-concentrators.

We can also use Theorem 2 as a benchmark to compare candidate designs of (N,M,C)-concentrators. Designs with Nx' crosspoints where x' is "close" to the value of x which satisfies (10) are at times acceptable for those cases when no design with Nx crosspoints is known. For example, consider the composite binomial network of Figure 4. This design is obtained by taking a $\binom{4}{2}$-network as shown in Figure 1 and replacing every crosspoint

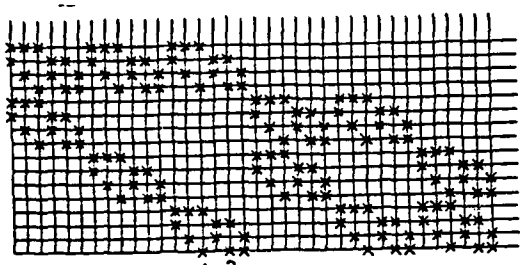


Fig. 4. A $(\frac{4}{2})^2$-network.

with another $(\frac{4}{2})$-network together with the proper input and output expansion. The resulting network, in this case, is called a $(\frac{4}{2})^2$-network; and it can be shown that it is a design of a (36,16,11)-concentrator [4]. The $(\frac{4}{2})^2$-network has $Nx' = (36)\cdot(4)=144$ crosspoints. From (10), we find that $x = 2.548$; therefore $Nx = 91.733$.

Linear Concentrators and Superconcentrators.

A (Nk,Mk,Ck)-concentrator for given N,M, and C is called a linear concentrator since the number of inputs, outputs, and the capacity grow linearly with k. Linear concentrators have been used as network components to prove the existence of certain superconcentrator designs [5],[6]. A superconcentrator is an interconnection network of somewhat more powerful connecting capability than that of a concentrator in the sense that in a superconcentrator there are N inputs and N outputs and for any specified subsets of $K \leq N$ inputs and outputs, there exists a disjoint path to connect each input to some output in the respective subsets. Hence, unlike a concentrator, for a superconcentrator both the input and output subsets to be connected can be specified and they can be of any size less than or equal to N. However, within these subsets the particular one-to-one connections cannot be specified. Pippenger [6] proved the existence of superconcentrators with at most 40N crosspoints. This proof was based upon a recursive construction using (6k,4k,3k)-concentrators which Pippenger proved existed with at most 36k crosspoints. In this section, we will establish a lower bound on the number of crosspoints in superconcentrator designs using Pippenger construction. This will be accomplished by establishing lower bounds on crosspoints for the linear concentrators it utilizes.

To begin, consider the more general case of (Nk,Mk,Ck)-concentrators. For this case, an approximation to (10) of Theorem 2 for sufficiently large k is

$$(\frac{Ck}{Mk})^x \cdot (Nk) \cdot (Ck-x) + Ck - C^2k^2 = 0 \quad . \tag{11}$$

Now, upon rearranging (11) and dividing by k^2, we can consider the limit of the result as k approaches infinity. That is,

$$\lim_{k\to\infty} \{\frac{k^2}{k^2}((\frac{C}{M})^x \cdot NC - C^2) - \frac{k}{k^2}((\frac{C}{M})^x \cdot Nx - C)\} = 0 \quad . \tag{12}$$

Clearly, (12) converges to

$$(\frac{C}{M})^x \cdot NC - C^2 = 0 \quad . \tag{13}$$

Taking logarithms yields

$$x = \frac{\log N - \log C}{\log M - \log C} \quad . \tag{14}$$

Equation (14) represents a lower bound on the average number of crosspoints in any (Nk,Mk,Ck)-concentrator design for large k.

Returning then to the issue of Pippenger's superconcentrator construction which was based on the recursive use of (6k,4k,3k)-concentrators with 6 crosspoints per input, it is seen from (14) that a lower bound on the number of crosspoints per input for such linear concentrators is $x = 2.40942$. When (6k,4k,3k)-concentrators with $(2.40942)\cdot(6k)$ crosspoints are used in Pippenger's superconcentrator construction, it can be seen that a lower bound on the number of crosspoints in the resulting superconcentrator is 17.4565N.

It is of interest to consider alternative linear concentrators because Pippenger's superconcentrator construction only requires the use of concentrators where the capacity is equal to one-half the number of inputs, but the ratio between the number of inputs and outputs has some flexibility. Let β correspond to this ratio in the sense that we will consider $(n, \beta n, \frac{1}{2}n)$-concentrators. Now, from (14),

$$x = \frac{\log n - \log \frac{n}{2}}{\log \beta n - \log \frac{n}{2}} = \frac{\log 2}{\log 2 + \log \beta} \quad , \tag{15}$$

then the number of crosspoints in Pippinger's construction can be expressed as

$$n \cdot (2 \cdot \frac{\log 2}{\log 2 + \log \beta} + 1) \cdot \frac{1}{1-\beta} \quad , \tag{16}$$

where $\frac{1}{2} < \beta < 1$.

Numerical analysis shows that (16) is minimized when $\beta = 0.70251$. From (15), this results in $x = 2.03835$. Then, if $(n, 0.70251n, \frac{1}{2}n)$-concentrators with 2.03835n crosspoints were used with Pippenger's construction, the result would be a lower bound of 17.0652N crosspoints. No design of a superconcentrator using Pippenger's construction could have less crosspoints.

References

1. M. Pinsker, "On the Complexity of a Concentrator," Proc. 75th International Teletraffic Conference Stockholm, 1973, pp.318/1-318/4.

2. N. Pippenger, "On the Complexity of Strictly Nonblocking Concentration Networks," IEEE Trans.Comm., Vol. COM-22, 1974, pp. 1890-1892.

3. G. Masson, "Binomial Switching Networks for Concentration and Distribution," IEEE Trans. Comm.,Vol. COM-25, No.9, 1977. pp. 873-883.

4. S. Nakamura and G. Masson, "Higher Order Composite Concentrators," Proc. 14th Ann. Conf. on Information Sciences and Systems, 1980.

5. L. Valiant, "On Non-linear Lower Bounds in Computational Complexity," Proc. 7th Ann. ACM Symp. on Theory of Computing, Albuquerque, N.M., 1975.

6. N. Pippenger, "Superconcentrators," SIAM Jour. Comp., Vol. 6, No. 2, 1977, 00. 298-304.

LOWER BOUNDS ON CROSSPOINTS IN CONCENTRATORS AND SUPERCONCENTRATORS

Shinji Nakamura and Gerald M. Masson
Department of Electrical Engineering
The Johns Hopkins University
Baltimore, Maryland 21218

Abstract

Lower bounds on the required number of crosspoints in realizations of interconnection networks called concentrators and superconcentrators are given. This work is in contrast to many of the other results in the literature which establish upper bounds by proving that with a certain number of crosspoints, various concentrator designs do exist. The lower bounds are obtained by using a straightforward necessary condition on the number of crosspoints for the special case of sparse crossbar full capacity concentrators. It is seen that this condition must be satisfied by all embedded full capacity concentrators contained in more general concentrator designs, and on the basis of this observation, a general necessary condition on the required number of crosspoints in a concentrator is established. This general crosspoint condition is exploited to obtain our lower bound results by demonstrating that an equal distribution of a given number of crosspoints between inputs and outputs in a sparse crossbar concentrator is the most efficient use of those crosspoints relative to maximizing the resulting capacity. As the use of sparse crossbar concentrators as components in the designs of other types of interconnection networks is common, our lower bound results can also be used to obtain lower bounds on crosspoints in those interconnection network designs as well. In particular Pinsker's multistage concentrators and Pippenger's superconcentrators are considered in this paper.

Introduction

An (n,m,c)-concentrator is an interconnection network with n inputs and m outputs $(m<n)$ where for an specified subset of $c \leq m$ inputs, all of the specified inputs can simultaneously be connected by means of disjoint paths to some set of c outputs. c is referred to as the capacity of the concentrator. The crucial limitation of the connecting capability of this type of interconnection network is that given a specified set of inputs, in order to simultaneously connect each input in this set to an output, the outputs cannot in general be arbitrarily specified (as would be the case in a permutation network).

The actual connections of inputs to outputs in a concentrator are achieved by means of switching elements referred to as crosspoints. The detailed implementation of an actual crosspoint is, of course, application dependent. In the following, a crosspoint can simply be viewed as a switching element which when "closed" provides a connecting path between the line entering it and the line leaving it, as opposed to when it is "open" and no such connecting path exists between these two lines.

A specification of the crosspoint placement between the inputs and outputs will be referred to as a design or explicit construction of a concentrator. Clearly, the simplest design of a concentrator is in the form of a so-called (sparse or complete) crossbar network, where between any input and any output there is at most one crosspoint. Concentrators can also be realized by designs in the form of composite networks. These are made up of various serial and parallel combinations of sparse or complete crossbar concentrators. Initially in this paper we will be concerned with only the sparse crossbar concentrator realizations, but, in the concluding sections, we will consider composite networks as well.

Obviously, a complete crossbar network in which there is a crosspoint between each of the n inputs and m outputs will function as an (n,m,m)-concentrator. This special case where $c=m$ will be referred to as a full capacity concentrator and will henceforth be denoted more simply as an (n,m)-concentrator. Similarly, a sparse crossbar network in which there are crosspoints from each input to any $c<m$ outputs will function as an (n,m,c)-concentrator. However, as might be anticipated on the basis of the relatively small number of input to output mappings that must be realized by a concentrator, the nm and nc resulting crosspoints, respectively, for each of these cases are usually unnecessarily high. Indeed, it is well established that (n,m,c)-concentrators and (n,m)-concentrators can be constructed with $O(n)$ crosspoints. For example, Pinsker [1] has shown that (n,m)-concentrators can be constructed with at most $29n$ crosspoints. This was accomplished by means of a probabilistic argument that demonstrated the existence (but not the explicit crosspoint placement) of some sparse crossbar concentrators that are components in a composite concentrator network. Similarly, Pippenger [2] used a similar argument to show the existence of sparse crossbar designs of $(n, \frac{2}{3}n, \frac{1}{2}n)$-concentrators with at most $6n$ crosspoints. These were then used as components in a composite design of an interconnection network called a superconcentrator. These types of results provide upper bounds on the total number of crosspoints required in the design of the associated concentrator.

There also have been contributions regarding explicit constructions of concentrators. Most notably, Gabber and Galil [3] have recently given a construction of a sparse crossbar concentrator which was first studied by Margulis [4] and which furthermore permits an explicit construction of a

superconcentrator in the form of the composite network suggested by Pippenger. Masson [5] also gave a construction of a sparse crossbar concentrator with what is referred to as a binomial crosspoint placement.

In this paper, we will be concerned with both the minimum number of crosspoints required in (n,m,c) and (n,m)-concentrators and some explicit constructions. Regarding the former, we will establish lower bounds for sparse crossbar concentrator designs. With these results, we will then be able to examine some explicit sparse crossbar constructions, seeing in some cases (for example, Masson's binomial concentrator) that these designs are, indeed, minimum. Finally, we will be able to determine lower bounds on the composite network realizations of concentrators given by Pinsker and superconcentrators given by Pippenger.

Preliminaries

To consider sparse crossbar networks as concentrator designs, we will use the following notation and definitions. Any sparse crossbar network, denoted in general as N, can be described as a triplet (I,O,R), where I is the input set, $|I| = n$; O is the output set, $|O| = m$; and R is a relation between I and O where for $i \in I$ and $o \in O$, $o \in R(i)$ implies there exists a crosspoint between input i and output o. This will at times be expressed in the following by the statement that input i is incident to output o, or, equivalently, that output o is incident to input i.

A subnetwork, N', of the sparse crossbar network, N, described by (I,O,R) is simply a network described by (I',O',R') where $I' \subseteq I$, $O' \subseteq O$, and $R' \subseteq R$.

The isolation set, $\sigma(O')$, of a subset $O' \subseteq O$ of a sparse network described by (I,O,R) is the subset $I' \subseteq I$ consisting of inputs incident to outputs only in O'. That is,

$$\sigma(O') = \{i \mid i \in I \text{ and } R(i) \subseteq O'\} .$$

The total number of crosspoints between the inputs of $\sigma(O')$ and the outputs of O' will be denoted as t'. It is clear that

$$t' = \sum_{j \in \sigma(O')} |R(j)| .$$

We will at times in the following consider concentrator designs where the explicit crosspoint placement is not specified, but, instead, only the crosspoint distribution of the design is given. $W(j)$ will denote the number of inputs in a sparse crossbar network that are incident to exactly j outputs. In the discrete case,

$$W(j) = |\{i \mid i \in I \text{ and } |R(i)| = j\}|$$

and

$$\sum_{j=1}^{m} W(j) = n .$$

This latter crosspoint distribution and any other such distribution where each input is incident to exactly the same number of outputs will be referred to in the following as a 1-point or singular crosspoint distribution. More generally an ℓ-point crosspoint distribution is one in which there is a set of ℓ distinct incidence values, $J = \{j_1, j_2, \ldots, j_\ell\}$, such that

$$W(j) \neq 0 \quad j \in J ,$$

$$W(j) = 0 \quad j \notin J ,$$

$j_i \neq j_k$ for all j_i and j_k in J, $i \neq k$,

and where

$$\sum_{j \in J} W(j) = n .$$

Full Capacity Sparse Crossbar Concentrators

The following is a necessary condition on the number of crosspoints in designs of full capacity concentrators.

Theorem 1: The minimum number of crosspoints required in a sparse crossbar design of an (n,m)-concentrator is $(n-m+1)m$.

Proof: Assume that some output, say $o \in O$ of a sparse crossbar design of an (n,m)-concentrator was incident to $n-m$ inputs. This means there are $n-(n-m) = m$ inputs with no crosspoints to o. But, clearly this contradicts the assumption that this design functions as a (n,m)-concentrator, since these m inputs only have crosspoints to, and, therefore, can be connected to, at most $m-1$ outputs. Hence, in a sparse crossbar (n,m)-concentrator design, each output must be incident to at least $n-m+1$ inputs. This results in a total of at least $(n-m+1)m$ crosspoints in any design.

Q.E.D.

A Lower Bound for Sparse Crossbar (n,m,c)-Concentrators

In the previous section, we showed minimum sparse crossbar designs of (n,m,c)-concentrators for the special case where $m = c$. In this section we will give a lower bound on the number of crosspoints required in any sparse crossbar (n,m,c)-concentrator, $c<m$. It will be seen in the following section that in some cases designs satisfying this lower bound can be given.

Consider any sparse crossbar design of an (n,m,c)-concentrator. Assume that this design contains nx_o crosspoints, where x_o is the incidence mean of the design. That is, ignoring for the moment that in general x_o is not an integer, each input, on the average, is incident to x_o outputs. In this sparse crossbar network, since by assumption the capacity is c, any choice of c of the m outputs, say, O', defines a subnetwork relative to the inputs in the isolation set, $\sigma(O')$, which must function as a full capacity concentrator. Let $\{O_1, O_2, \ldots, O_{\binom{m}{c}}\}$ be the $\binom{m}{c}$ possible choices of output sets of size c. Consider $\sigma(O_i)$, $i=1,\ldots,\binom{m}{c}$. It follows that

$$(|\sigma(O_i)| - c + 1) \cdot c \leq t_i . \qquad (1)$$

Summing over all $\binom{m}{c}$ choices of output sets of size c, and dividing by $\binom{m}{c}$ yields

$$\frac{\sum_{i=1}^{\binom{m}{c}} |\sigma(O_i)|}{\binom{m}{c}} - c+1 \leq \frac{\sum_{i=1}^{\binom{m}{c}} t_i}{\binom{m}{c}} . \qquad (2)$$

Recall that $W(j)$ is the number of inputs in a sparse crossbar network that are incident to exactly j outputs. Now suppose that a given input is incident to j outputs. It follows that of the $\binom{m}{c}$ possible selections of c of the m outputs, exactly $\binom{m-j}{c-j}$ of those choices isolate this input. It is furthermore easily seen that

$$\binom{m-j}{c-j} = \binom{m}{c} \cdot \frac{\binom{c}{j}}{\binom{m}{j}} = \binom{m}{c} \cdot \frac{c!\,(m-j)!}{m!\,(c-j)!} .$$

For convenience in the following, we will let

$$g(j) = \frac{\binom{c}{j}}{\binom{m}{j}}$$

so that

$$\binom{m-j}{c-j} = \binom{m}{c} g(j) .$$

With the above and the crosspoint distribution, it follows that

$$\sum_{i=1}^{\binom{m}{c}} |\sigma(O_i)| = \sum_{j=1}^{m} \binom{m}{c} g(j) W(j) ,$$

or

$$\frac{\sum_{i=1}^{\binom{m}{c}} |\sigma(O_i)|}{\binom{m}{c}} = \sum_{j=1}^{m} g(j) \cdot W(j) .$$

Simlarly, it can be seen that

$$\frac{\sum_{i=1}^{\binom{m}{c}} t_i}{\binom{m}{c}} = \sum_{j=1}^{m} g(j) \cdot W(j) \cdot j .$$

Using the above in (2) and rearranging terms yields the crosspoint distribution form of our necessary condition on sparse crossbar designs of (n,m,c)-concentrators:

$$\sum_{j=1}^{m} (g(j) \cdot W(j) \cdot (c-j)) - c^2 + c \leq 0 . \qquad (3)$$

To obtain our lower bound, we now must consider all possible crosspoint distributions that satisfy (3) for sparse crossbar designs of (n,m,c)-concentrators. For all practical purposes this task would be impossible were it not for an intuitively satisfying property of crosspoint distributions that reflects itself into the summation terms of (3). Given some fixed number of crosspoints, say, nx, $x<c$, which are to be placed between the n inputs and m outputs of a crossbar grid so as to provide the maximum possible capacity, there is a natural tendency, because of analogies which can be made with other physical phenomena, to in general consider crosspoint distributions that evenly distribute the crosspoints among the inputs. In other words, intuitively one would expect to be able to maximize the capacity with a singular crosspoint distribution. Indeed, if we ignore the fact that x in general is not an integer, this turns out to be the proper approach, and this is demonstrated rigorously in [7]. Hence we can conclude that to obtain our lower bound on the total number of crosspoints required in sparse crossbar designs of (n,m,c)-concentrators from our necessary condition as given by (3), it is only necessary to consider the singular crosspoint distribution case. This will be formally stated as

Theorem 2: A lower bound on the number of crosspoints in a sparse crossbar design of an (n,m,c)-concentrator is nx where x satisfies

$$\frac{\binom{c}{x}}{\binom{m}{x}} \cdot n \cdot (c-x) - c^2 + c = 0 . \qquad (4)$$

Binomial Networks

An obvious question raised by the lower bound equation of Theorem 2 is whether or not sparse crossbar designs actually satisfying the equation exist. It can be seen that when the values of the parameters n, m, and c satisfy certain binomial interrelationships, this is the case. More particularly, an $\binom{m}{v}$-network has $\binom{m}{v}$ inputs, m outputs, and a singular crosspoint distribution wherein each input is incident to v outputs. Moreover in such a network, for any input, the placement of the v crosspoints between it and the outputs is distinct from that of any other input. The capacity of such a design is known to be $v+2$ [5]. Substituting $n = \binom{m}{v}$, $x = v$, and $c = v+2$ into the left-hand side of (4) yields

$$\frac{\binom{v+2}{v}}{\binom{m}{v}} \cdot \binom{m}{v} \cdot (v+2-v) - (v+2)^2 + (v+2)$$

$$= \frac{(v+2)(v+1)}{2} \cdot (2) - (v+2)^2 + (v+2) = 0 .$$

In other words, $x = v$ satisfies (4) relative to $(\binom{m}{v}, m, v+2)$-concentrators, and binomial networks are, therefore, minimum sparse crossbar designs of such concentrators.

Fixed Ratio Concentrators

For given integers n, m, and c, an (nk,mk,ck)-concentrator will be referred to as a fixed ratio concentrator since, although the number of inputs, outputs, and the capacity vary according to k, their ratios remain fixed. For fixed ratio concentrators, we can again use (4) to determine a lower bound on the average number of crosspoints in their sparse crossbar designs. For this case

with sufficiently large k, it is easy to see that our lower bound equation (4) can be approximated as

$$\left(\frac{ck}{mx}\right)^{x} \cdot nk\ (ck - x) - c^2k^2 + ck = 0 ,$$

or

$$k^2\left(\left(\frac{c}{m}\right)^{x} \cdot nc - c^2\right) - k\left(\left(\frac{c}{m}\right)^{x} \cdot nx - c\right) = 0 .$$

Dividing through by k^2 and taking the limit as k approaches infinity yields

$$\lim_{k\to\infty}\left(\frac{k^2}{k^2}\left(\left(\frac{c}{m}\right)^{x} \cdot nc - c^2\right) - \frac{k}{k^2}\left(\left(\frac{c}{m}\right)^{x} \cdot nx - c\right)\right) = 0 ,$$

or

$$\left(\frac{c}{m}\right)^{x} \cdot nc - c^2 = 0 .$$

Taking logarithms, we have

$$x = (\log n - \log c)/(\log m - \log c) . \qquad (5)$$

Equation (5) represents a lower bound on the average number of crosspoints in any sparse crossbar design of an (nk,mk,ck)-concentrator for large k.

Pinsker's Concentrators

In this section we will give lower bounds on some of the prominent composite networks for concentrators. These composite networks use sparse crossbar networks as their building blocks or components, and, hence, their lower bounds will be based primarily upon the results of the previous sections on sparse crossbar concentrators. To expidite the presentation of these results (as well as the results of the following section), we will first develop some notation to describe composite networks.

Given two (sparse crossbar or composite) networks N_1 and N_2 where N_1 has an input set I_1 and, an output set O_1, and where N_2 has an input set I_2 and an output set O_2, $|O_1| = |I_2|$, then $N_1 \Rightarrow N_2$ *will denote the serial product* of N_1 and N_2 and will represent a composite network having I_1 for an input set and O_2 for an output set, and where the output of N_1 and the inputs of N_2 are identified or associated on a one-to-one basis.

Given two (sparse crossbar or composite) networks N_1 and N_2 where $|I_1| = n_1$ and $|O_1| = m_1$ and where $|I_2| = n_2$ and $|O_2| = m_2$, then $N_1|a,b|N_2$, where $a<\min(n_1,n_2)$, $b<\min(m_1,m_2)$ will denote the parallel product of N_1 and N_2 and will represent a composite network having n_1+n_2-a inputs formed from $I_1 \cup I_2$ but where the "lower" a inputs of N_1 and the "upper" a inputs of N_2 are superimposed so as to have paths to the outputs available either through N_1 or N_2, and having m_1+m_2-b outputs formed from $O_1 \cup O_2$ but where the "lower" b outputs on N_1 and the "upper" b outputs of N_2 are superimposed.

Given a network N, then $\bar{N}$ will denote the inverse of that network where the inputs of N are used as outputs and vice-versa.

Finally, we will let I_n denote the n input, n output sparse crossbar network with crosspoints only on the main diagonal.

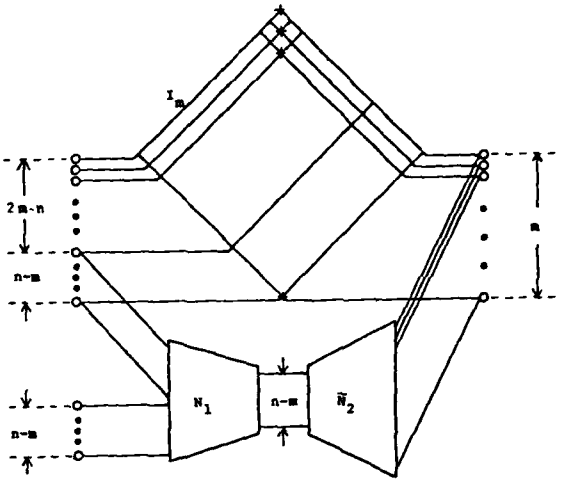


Figure 1: A composite (n,m)-concentrator.

Now, consider the composite network shown in Figure 1. This composite network was first presented by Pinsker [1]. It can be shown [1] that if

N_1 is an $(2(n-m)\ ,\ n-m)$ - concentrator,

N_2 is an $(m,n-m)$ - concentrator,

and $2m \geq n$,

then this composite network is an (n,m) concentrator. Clearly, this network can be described as

$$I_m\,|(n-m)\ ,\ m|\,(N_1 \Rightarrow \bar{N}_2) .$$

Pinsker used a composite network of this type to show that in general (n,m)-concentrators existed with at most $29n$ crosspoints. This was accomplished by considering in particular a composite network design of an $(n,\frac{5}{6}n)$-concentrator as described by

$$I_{\frac{5}{6}n}\ |\tfrac{1}{6}n\ ,\ \tfrac{5}{6}n\,|\,(N_1 \Rightarrow \bar{N}_2)$$

where $(N_1 \Rightarrow \bar{N}_2)$ is a composite design in which:

(i) N_1 is a $(\frac{1}{3}n\ ,\ \frac{1}{6}n)$-concentrator in the form of the composite network

$$(N_{11} \Rightarrow N_{12})$$

in which N_{11} is a $(\frac{1}{3}n\ ,\ \frac{1}{5}n\ ,\ \frac{1}{6}n)$-fixed ratio concentrator and N_{12} is $(\frac{1}{5}n\ ,\ \frac{1}{6}n)$-concentrator (which it should be noted has an input set, output set, and capacity equal to $\frac{1}{5}$ that of the overall concentrator);

(ii) N_2 is a $(\frac{5}{6}n\ ,\ \frac{1}{6}n)$-concentrator in the form of the composite network

$$(N_{23} \Rightarrow N_{22} \Rightarrow N_{21})$$

in which N_{23} is a $(\frac{5}{6}n\ ,\ \frac{1}{6}n)$-fixed ratio concentrator, N_{22} is a $(\frac{1}{2}n\ ,\ \frac{1}{5}n\ ,\ \frac{1}{6}n)$-fixed ratio concentrator, and N_{21} is a

$(\frac{1}{5}n, \frac{1}{6}n)$-concentrator (which, again, it should be noted has an input set, output set, and capacity equal to $\frac{1}{5}$ that of the overall concentrator).

Pinsker's design is clearly recursive. Hence, using (5) for the component fixed ratio concentrators, we have that a lower bound on the number of crosspoints in N_{11} and N_{22} is

$$\frac{1}{3}n\left(\frac{\log\frac{1}{3} - \log\frac{1}{6}}{\log\frac{1}{5} - \log\frac{1}{6}}\right) = 1.26726\,n$$

and that a lower bound on the number of crosspoints in N_{23} is

$$\frac{5}{6}n\left(\frac{\log\frac{5}{6} - \log\frac{1}{6}}{\log\frac{1}{3} - \log\frac{1}{6}}\right) = 1.93494\,n\,.$$

Thus, a lower bound on the number of crosspoints in Pinsker's composite design of a $(n, \frac{5}{6}n)$-concentrator is

$$(\tfrac{5}{6}n) + 1.93494\,n + 2(1.26726\,n) + (\text{lower bound for a Pinsker } (\tfrac{1}{5}n, \tfrac{1}{6}n)\text{-concentrator}). \quad (6)$$

Recursive evaluation (6) yields a lower bound on the number of crosspoints in a Pinsker $(n, \frac{5}{6}n)$-concentrator to be $7.4491\,n$. Pinsker, it should be noted, showed with his design the existence of $(n, \frac{5}{6}n)$-concentrators with at most $22\,n$ crosspoints.

Now, with the design for $(n, \frac{5}{6}n)$-concentrators, Pinsker considered $(n, \frac{n}{a})$-concentrators for $1 < a \leq n$. This was done by considering ranges of a. We will likewise consider our lower bound over these ranges. Figure 2 shows a graph of k versus a where kn is our lower bound on Pinsker $(n, \frac{n}{a})$-concentrators.

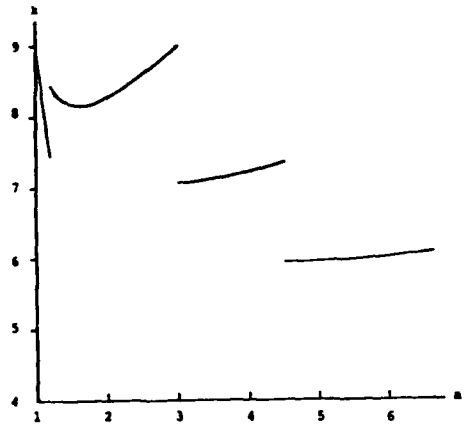


Figure 2. Lower bound on Pinsker's $(n, \frac{n}{a})$-concentrator.

Range (i) $1 < a \leq \frac{6}{5}$: In this range, the above $(n, \frac{5}{6}n)$-concentrator is used by simply designing this network to have the correct number of $\frac{n}{a}$ outputs and then scaling down the number of inputs appropriately. Hence, in this range, the lower bound on the number of crosspoints in Pinsker's $(n, \frac{n}{a})$-concentrator is $(7.4491)(\frac{6}{5}\cdot\frac{n}{a})$

Range (ii) $\frac{6}{5} \leq a \leq 3$: In this range, Pinsker uses the serial product of a fixed ratio $(n, \frac{6}{5}\frac{n}{a}, \frac{n}{a})$-concentrator and a $(\frac{6}{5}\frac{n}{a}, \frac{n}{a})$-concentrator. Hence, in this range our lower bound on the number of crosspoints in Pinsker's $(n, \frac{n}{a})$-concentrator is

$$\left(\frac{\log a}{\log\frac{6}{5a} + \log a}\right) n + (7.4491)\left(\frac{6}{5}\frac{n}{a}\right)\,.$$

Range (iii) $3 \leq a \leq n$: This range is treated by Pinsker as a set of subranges. In each subrange, his design consists of an initial serial product of fixed ratio concentrators to reduce the required concentration to that which can be addressed by a concentrator in range (ii). A serial product of that initial composite network is then taken with the appropriate concentrator from range (ii) to complete the design. For example, for the subrange $3 \leq a \leq \frac{9}{2}$, a serial product of a fixed ratio $(n, \frac{2}{3}n, \frac{n}{a})$-concentrator and an $(\frac{2}{3}n, \frac{n}{a})$-concentrator with a design from range (ii) is used. Hence, our lower bound on crosspoints on Pinsker's $(n, \frac{n}{a})$-concentrator in this subrange is

$$\left(\frac{\log a}{\log\frac{2}{3} + \log a}\right) n + \left(\frac{\log a}{\log\frac{6}{5a} + \log a}\right)\cdot\frac{2}{3}n + (7.4491)\,\frac{6}{5}\cdot\frac{2}{3}\cdot\frac{n}{a}\,.$$

Likewise, in the subrange $\frac{9}{2} \leq a \leq \frac{27}{4}$, the serial product of the fixed ratio $(n, \frac{2}{3}n, \frac{n}{a})$-concentrator and a fixed ratio $(\frac{2}{3}n, \frac{4}{9}n, \frac{n}{a})$-concentrator is used to bring the concentrator into range (ii). The serial product of this initial network is then taken with a $(\frac{4}{9}n, \frac{n}{a})$-concentrator resulting in a lower bound of

$$\left(\frac{\log a}{\log\frac{2}{3} + \log a}\right) n + \left(\frac{\log\frac{2}{3} + \log a}{\log\frac{4}{9} + \log a}\right)\frac{2}{3}n + \left(\frac{\log\frac{4}{9} + \log a}{\log\frac{6}{5a} + \log a}\right)\frac{4}{9}n + (7.4491)\left(\frac{6}{5}\cdot\frac{4}{9}\,n\right)\,.$$

Similarly, the subrange $\frac{27}{4} \leq a \leq \frac{81}{8}$ uses the serial product of 3 fixed ratio concentrators, namely, an $(n, \frac{2}{3}n, \frac{n}{a})$-concentrator, an $(\frac{2}{3}n, \frac{4}{9}n, \frac{n}{a})$-concentrator, and an $(\frac{4}{9}n, \frac{8}{27}n, \frac{n}{a})$-concentrator, to bring the concentrator into range (ii). Hence, our lower bound curve of Pinsker's $(n, \frac{n}{a})$-concentrator can be obtained.

Pippenger's Superconcentrator

An (n,n)-superconcentrator is an interconnection network of somewhat more powerful connecting capability than that of a concentrator in the sense that there are n inputs and n outputs and for any specified subsets of $k < n$ inputs and outputs, there exists a disjoint path to connect each input to some output in the respective subsets. Hence, unlike a concentrator, for a superconcentrator both the input and output sub-

sets to be connected can be specified and they can be of any size less than or equal to n. However, within these subsets the particular one-to-one connections cannot be specified.

(n,n)-superconcentrators were defined by Valiant [8] and shown to exist with at most 238 n crosspoints. Pippenger later improved this bound to 40 n [2]. Recently, Gabber and Galil [3] gave an explicit construction of an (n,n)-superconcentrator requiring 273 n crosspoints. We will conclude this paper by considering a lower bound on Pippenger's superconcentrator design.

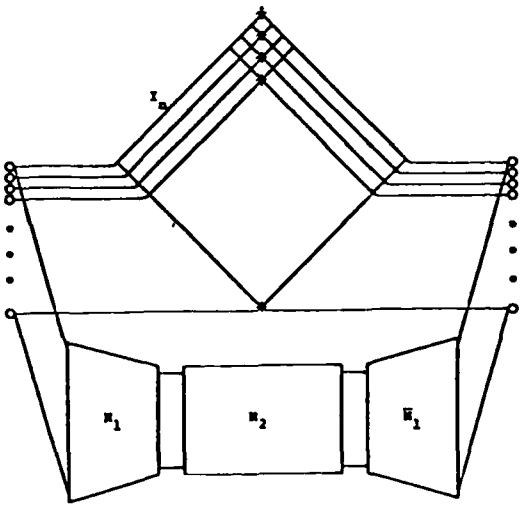


Figure 3: Pippenger's (n,n)-superconcentrator.

Consider the composite network shown in Figure 3. This network can be described as

$$I_n \,|n,n|\, (N_1 \Rightarrow N_2 \Rightarrow \bar{N}_1) .$$

Pippenger showed that if

N_1 is an $(n, \frac{2}{3}n, \frac{1}{2}n)$-concentrator

and

N_2 is an $(\frac{2}{3}n, \frac{2}{3}n)$-superconcentrator,

then this composite network is an (n,n)-superconcentrator. Hence, a lower bound on Pippenger's (n,n)-superconcentrator at this point is $N + 2(\frac{\log 2}{\log 4 - \log 7})n +$ (lower bound on Pippenger's $(\frac{2}{3}n, \frac{2}{3}n)$-superconcentrator). Recursive evaluation yields a lower bound on crosspoints in a Pippenger (n,n)-superconcentrator as 17.4565 n.

The key requirement in Pippenger's design is that N_1 has a capacity of $\frac{1}{2}n$, but the only requirement on the size of the output set, say, βn is that $1 < \beta < \frac{1}{2}$. Our lower bound on this fixed ratio concentrator is

$$(\frac{\log 2}{\log 2 + \log \beta}) n .$$

This gives the number of crosspoints in Pippenger's (n,n)-superconcentrator as

$$(1 + 2(\frac{\log 2}{\log 2 + \log \beta})) \cdot (\frac{1}{1-\beta}) n$$

which has a minimum of 17.0652 n at $\beta = 0.70251$. In this version of Pippenger's design, $(n, 0.70251\,n, \frac{1}{2}n)$-concentrators are used, each with an average of 2.03835 crosspoints per input. Pippenger's (n,n)-superconcentrator cannot be constructed with fewer crosspoints.

Conclusion

In this paper we have used a straightforward requirement on the number of crosspoints in sparse crossbar (n,m)-concentrators to obtain lower bounds for the more general cases of (n,m,c)-concentrators. These lower bounds were attained by means of an argument which showed that a non-biased or singular distribution of a given number of crosspoints between inputs and outputs in a sparse crossbar network is the most efficient use of those crosspoints regarding the resulting capacity. In certain cases it was seen that the lower bound could actually be satisfied: namely, when the number of inputs, outputs, and the capacity satisfy the so-called binomial interrelationships, as in such situations the singular distribution satisfying our lower bound requirements is achievable with an integer number of crosspoints per input. The use of sparse crossbar concentrators as components in the designs of other types of interconnection networks permits us to use our lower bound results to obtain lower bounds on crosspoints in those designs as well. In particular we considered Pinsker's multi-stage concentrators and Pippenger's superconcentrators.

References

1. M. Pinsker, "On the Complexity of a Concentrator," Proc. 75th International Teletraffic Conference, Stockholm, 1973, pp.318/1-318/4.

2. N. Pippenger, "Superconcentrators," SIAM Jour. Comp., Vol. 6, No. 2, 1977, pp.298-304.

3. O. Gabber and Z. Galil, "Explicit Constructions of Linear Size Superconcentrators," to appear in the Journal of Computer and System Sciences.

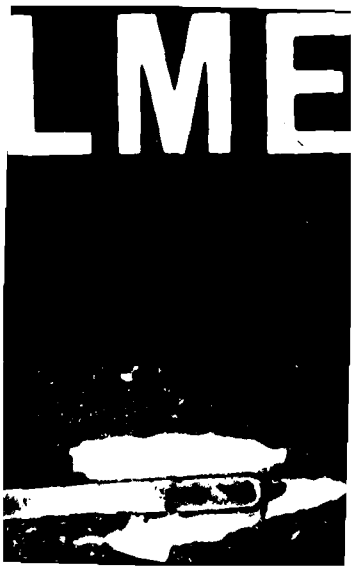
4. G. A. Margulis, "Explicit Constructions of Concentrators," Problems of Information Transmission, Plenum, New York, 1975.

5. G. Masson, "Binomial Switching Networks for Concentration and Distribution," IEEE Trans. Comm., Vol. COM-25, No. 9, 1977, pp.873-883.

6. P. Hall, "On Representatives of Subsets," J. London Math. Soc., Vol. 10, 1935, pp.26-30.

7. S. Nakamura, "Lower Bounds and Product Designs for Concentrators," Ph.D. Dissertation, The Johns Hopkins University, in preparation.

8. L. Valiant, "On Non-linear Lower Bounds in Computational Complexity," Proc. 7th Ann. ACM Symp. on Theory of Computing, Albuquerque, N.M., 1975.


LME